SOME CONTRIBUTIONS TO DIMENSIONALITY REDUCTION

By

Wei Tong

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Computer Science

2010

ABSTRACT

SOME CONTRIBUTIONS TO DIMENSIONALITY REDUCTION

By

Wei Tong

Dimensionality reduction is a long standing challenging problem in the fields of statistical learning, pattern recognition and computer vision. Numerous algorithms have been proposed and studied in the past decades. In this dissertation we address several challenging problems emerged in recent studies of dimensionality reduction. We first explore the dimensionality reduction method for semi-supervised classification via the idea of mixed label propagation in which we attempt to find the best one dimensional embedding of the data in which data points in different classes can be well separated and the class labels are obtained by simply thresholding the one dimensional representation. In the next, we explore the dimensionality reduction methods for non-vector data representations. We first look into the problem in which a datum is represented by a matrix. We give a convex formulation to the problem of dimensionality reduction for matrices and developed an efficient approximating algorithm to solve the associated semi-definite programming problem. In the last, we studied the problem of dimensionality reduction with even more challenging data representation, i.e., each datum is described as a different number of unordered vectors. We convert the problem into the functional data dimensionality reduction and study it in the context of large scale image retrieval.

ACKNOWLEDGMENTS

Time flies! It has been six years since I started my Ph. D. study at MSU. Looking back, I am surprised and at the same time very grateful for all I have received throughout these years.

My first, and most earnest, acknowledgment must go to my advisor and the chair of my Committee Dr. Rong Jin. With his enthusiasm, his inspiration, and his great efforts to explain things clearly and simply, he helped to make mathematics easy for me. Throughout my thesis-writing period, he provided encouragement, sound advice, good teaching, good company, and lots of good ideas. I would have been lost without him. In every sense, none of this work would have been possible without him. Many thanks also to committee members Dr. Anil Jain, Dr. Joyce Chai and Dr. Selin Aviyente.

Many thanks to my previous advisor Dr. Juyang Weng with whom I worked in my first two years of study. I would like to present my sincere appreciation for his valuable support during those years.

Many people on the faculty and staff of the CSE department at MSU assisted and encouraged me in various ways during my studies. I am especially grateful to Prof. George Stockman, Dr. Eric Torng, Linda Moore, Norma Teague, Adam Pitcher and Debbie Kruch.

My Ph. D. study would not have been the same without the great support from my friends from EI, PRIP and LINKS labs. I am particularly thankful to Xiao Huang, Nan Zhang, Shuqing Zeng, Zhengping Ji, Feilong Chen, Xiaoguang Lv, Hong Chen, Yi Chen, Yi Liu, Tianbao Yang, Yang Zhou, Jinfeng Yi and Fengjie Li.

I also owe a huge debt of gratitude to all the people from the Institute of Water Research at MSU where I worked as a system administrator for more than three years. They are, Jon Bartholic, Jeremiah Asher, Yi Shi, Cindy Brewbaker, Stephanie Smith, Glenn O'Neil, Debra Porter, Ruth Kline-Robach.

My final, and the most heartfelt, acknowledgment must go to my family for their unlimited and unconditional encouragement, support and love. Learning to love you and to receive your love make me a better person. You have my everlasting love!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

The curse of dimensionality[8] is a well known problem in the fields of machine learning and statistical pattern recognition. It refers to the fact that, in the absence of simplified assumption, the sample size needed to estimate a function of multiple variables to a given degree of accuracy grows exponentially in the number of variables. In other words, when the number of variables – the dimensionality of data – increases, the number of data points needed to determine an accurate estimation could quickly exceed one's processing ability.

To illustrate the challenge of handling high dimensional data, consider a face recognition system based on grey scale images. Assume the size of each image is 256×256, which can also be treated as a vector of 65536 dimension. In order to train a perceptron, a simple linear classifier for face recognition, it requires solving an optimization problem with 65537 parameters. The size of the recognition problem will increase exponentially when considering a multi-layer perceptron, making it impractical unless certain simplifying assumptions are made.

The challenge of handling high dimensional data calls for the techniques of dimensionality reduction, whose goal is to significantly reduce the number of features that are used for data description. More specifically, suppose we have a high dimensional data set $D = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$, where $\mathbf{x}_i \in \mathbb{R}^m$, $i = 1, \cdots n$, are input patterns of m features. The objective of dimensionality reduction is to compute n output patterns $\mathbf{y}_i \in \mathbb{R}^r$, one for each data point in D, that provide a "faithful" low dimensional representation of the original data set with $r \ll m$. By faithful, we mean that the mapping from $\mathbf{x}$ to $\mathbf{y}$ preserves the geometric relationship among data points. Namely, if two data points are close to each other in the

original space $\mathbb{R}^m$ measured by the Euclidean distance, they should also be close to each other in the mapped space $\mathbb{R}^r$, and vice versa. Without loss of generality we assume, if not specified, the input patterns are centered at the origin, i.e., $\sum_{i=1}^{n} \mathbf{x}_i = \mathbf{0}$.

Numerous algorithms have been proposed and studied in the past decades, for example, the classical PCA family algorithms, graph based methods and matrix factorization based methods. However, there are two limitations of these methods. The first one is that most of these methods are either supervised or unsupervised methods, there are few algorithms developed for semi-supervised dimensionality reduction in which information from both labeled and unlabeled data can be explored. In this dissertation, we try to develop such an algorithm for semi-supervised dimensionality reduction via the idea of mixed label propagation. We approximate a data classification problem into a dimensionality reduction one by finding the best one dimensional embedding of the data in which data points in different classes can be well separated. We then obtain the class labels by simply thresholding the one dimensional representation.

The second limitation of the existing dimensionality reduction methods is that each input data must be a vector with fixed length. However, in many real applications, the natural representation of each datum could be versatile. For example, an image is represented as a matrix of pixels, a text document can be represented as a set of vectors with different length. In those situations, in order to apply the dimensionality reduction methods, one has to convert the natural representation of each datum into the vector representation and this conversion may cause serious problems. For instance, by converting an image from the matrix presentation into the vector representation, all the neighborhood information between pixels are lost. This also makes the dimension of the resulting vector to the extremely high and often leads to very high computational cost when applying dimensionality reduction methods. In order to overcome this limitation, we study the problem of dimensionality reduction when each datum has a non-vector representation. More specifically, We first look into the problem in which a datum is represented by a matrix. We give a convex formulation to the problem

of dimensionality reduction for matrices and developed an efficient approximating algorithm to solve the associated semi-definite programming problem. Second, we studied the problem of dimensionality reduction with each datum is described as a different number of unordered vectors. We convert the problem into the functional data dimensionality reduction and study it in the context of large scale image retrieval.

The rest of the dissertation is organized as the following: in the next chapter, we give an literature review of the methods for dimensionality reduction which have close relationship to the problem we are going to solve in this dissertation. In the third chapter, we address the problem of dimensionality reduction for semi-supervised learning by mixed label propagation. In the fourth chapter, we study the problem when an input data is represented as a matrix instead of a vector. In the fifth chapter, we address the problem of how to reduced the dimensionality for functional data representation. The conclusion is presented in the last chapter.

CHAPTER 2

Literature Review of Dimensionality Reduction

In this chapter we start our review in Section 2.1 with one of the most widely used methods for dimensionality reduction, principal component analysis (PCA), followed by metric multidimensional scaling (MDS) and several unsupervised dimensionality reduction algorithms that are closely related to PCA and MDS. Although these classical methods were traditionally classified as a linear dimensionality reduction that transforms the input patterns by a linear projection, they can be extended to nonlinear dimensionality reduction by the introduction of kernel functions. Section 2.2 describes several graph-based methods for dimensionality reduction. They are non-linear dimensionality reduction methods and are particularly useful for analyzing high dimensional data that are sampled from a low dimensional manifold. Although these methods are based on different geometric intuitions and use different computational procedures, they share similar computational structures. Section 2.3 reviews dimensionality reduction methods based on matrix factorization that are not covered in most recent surveys of dimensionality reduction [135, 19, 45, 21]. In Section 2.4 we review both supervised and semi-supervised methods for dimensionality reduction. Lastly, in Section 2.5, we review the functional dimensionality reduction in which the data are presented by functions instead of vectors.

2.1 PCA and Related Methods

Principal component analysis (PCA) and its related linear methods, such as multidimensional scaling (MDS), are possibly the most widely used dimensionality reduction techniques. These algorithms are computationally efficient and easily implemented. As a result, these techniques have found a large number of applications as well as a number of different derivations. It is also important to note that despite the simplicity, the basic geometric intuition of PCA and MDS has inspired many algorithms for non-linear dimensionality reduction.

2.1.1 Principle Components Analysis (PCA)

Principle Components Analysis [67, 72] is an unsupervised dimensionality reduction method. The key idea of PCA is to identify the low dimensional representation of a high dimensional data set that most faithfully preserves its covariance structure. PCA has been referenced by many names in different fields. For example, in statistics and signal processing it is also known as the Karhunen-Loève transformation, the Hotelling transformation, and the empirical orthogonal function(EOF). Below we discuss several ways of deriving the algorithm of PCA based on different principles.

Reconstruction with Minimum Squared Error

Given a data set $D = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$, where $\mathbf{x}_i \in \mathbb{R}^m, i = 1, \cdots n$, PCA projects the input data $\mathbf{x}_i$ onto a r-dimensional subspace such that the reconstruction error is minimized. In other words, we need to find a projection matrix P with rank $r < m$ that minimizes the following objective function:

$$\min_{P} \quad \sum_{i=1}^{n} \|\mathbf{x}_i - P\mathbf{x}_i\|^2 \tag{2.1}$$

The optimal projection matrix can be factorized as $P = UU^\top$, where $U \in \mathbb{R}^{m \times r}$ has orthogonal columns. The r-dimensional representations are given as:

$$\mathbf{y}_i = U^\top \mathbf{x}_i, \quad i = 1, \cdots, n \tag{2.2}$$

The solution to (2.1) is obtained from the eigen-decomposition of the covariance matrix $C = \sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top$. Let C be expressed as $C = \sum_{i=1}^{m} \lambda_i \mathbf{u}_i \mathbf{u}_i^\top$, where λ_i is the i-th largest eigenvalue of C and $\mathbf{u}_i$ is the associated eigenvector. The optimal low dimensional representation can be computed using the equation(2.2) with $U = [\mathbf{u}_1 \cdots \mathbf{u}_r]$.

Finding an Informative Direction

PCA can also be derived from the point of view of finding the most informative direction [19]. Suppose we aim to find a direction $\mathbf{u} \in \mathbb{R}^m$ such that the projection $\mathbf{x}_i \cdot \mathbf{u}$ gives a good one dimensional representation of the original data (i.e., the projection should lose as little information of the original data as possible). Assume that the data in fact lie along a line embedded in $\mathbb{R}^m$, i.e., $\mathbf{x}_i = \boldsymbol{\mu} + \theta_i \mathbf{v}$, where $\boldsymbol{\mu}$ is the mean of the samples, $\mathbf{v} \in \mathbb{R}^m$ is a vector with unit length and $\theta_i \in \mathbb{R}$. The sample variance of the projection along $\mathbf{v}$ is

$$\text{var}_{\mathbf{v}} = \frac{1}{n} \sum_{i=1}^{n} ((\mathbf{x}_i - \boldsymbol{\mu}) \cdot \mathbf{v})^2 = \frac{1}{n} \sum_{i=1}^{n} \theta_i^2$$

The variance along other unit direction $\mathbf{v}'$ is

$$\text{var}_{\mathbf{v}'} = \frac{1}{n} \sum_{i=1}^{n} \left((\mathbf{x}_i - \boldsymbol{\mu}) \cdot \mathbf{v}'\right)^2 = \frac{1}{n} \sum_{i=1}^{n} \theta_i^2 \left(\mathbf{v} \cdot \mathbf{v}'\right)$$

Since $\left(\mathbf{v} \cdot \mathbf{v}'\right)^2 = \cos^2 \phi$, where ϕ is the angle between the two directions $\mathbf{v}$ and $\mathbf{v}'$, projected variance is maximized if and only if $\mathbf{v}' = \mathbf{v}$. Therefore, the most informative direction for the projection is the one for which the projected variance is maximized.

It can be shown that the problem in (2.1) is equivalent to the following optimization problem:

$$\begin{aligned} \max_{\mathbf{y}_i} \quad & \sum_{i=1}^{n} \|\mathbf{y}_i\|^2 = \frac{1}{2n} \|\mathbf{y}_i - \mathbf{y}_j\|^2 \\ \text{s. t.} \quad & U^\top U = I, \quad \mathbf{y}_i = U^\top \mathbf{x}_i \end{aligned} \tag{2.3}$$

which is equivalent to finding the projection for which the projected variance is maximized.

PCA Decorrelates the Samples

If we use the full set of orthonormal eigenvectors as the basis, then in this basis the data are represented as $Y = U^\top X$, where U is the $m \times m$ matrix obtained in Section 2.1.1. Thus, the sample covariance matrix in the new space C_Y is:

$$\begin{aligned} C_Y &= \frac{1}{n}\sum_{i=1}^{n} \mathbf{y}_i \mathbf{y}_i^\top \\ &= \frac{1}{n}\sum_{i=1}^{n} \left(U^\top \mathbf{x}_i\right)\left(U^\top \mathbf{x}_i\right)^\top \\ &= U^\top C_X U \\ &= \mathrm{diag}\left(\lambda_1, \cdots, \lambda_m\right) \end{aligned}$$

which means in the new basis, the sample covariance matrix is diagonalized. This implies the samples are uncorrelated in the space specified by matrix U. It is worth emphasizing that the notion of correlation is basis-dependent which is to say that data can be correlated in one basis but uncorrelated in another.

Limitation of PCA and Its Extensions

Although PCA is proved to be efficient and effective for dimensionality reduction in many applications, it however has its own limitations.

The first limitation of PCA is that it is a linear dimensionality reduction method and only identifies the linear subspace that best preserves the covariance structure of data. However, in many real world applications data essentially lie in nonlinear manifold whose structure is usually difficult to be discovered by PCA. A number of extensions have been developed to adapt PCA for non-linear dimensionality reduction. The most widely used approach is kernel PCA [136, 137], which extends the linear PCA by the introduction of kernel functions. We will discuss kernel PCA in Section 2.1.5.

Another limitation of PCA is the interpretation of the principal components can be difficult at times. Although the dimensions identified by PCA are uncorrelated variables constructed

as linear combination of the original variables and have some desirable properties, they do not necessarily correspond to meaningful physical quantities. In some cases, such loss of interpretability is unsatisfactory to the domain scientists. Non-negative matrix factorization [87] was introduced to give meaningful approximation of a non-negative matrix by non-negative low rank factorization. We will discuss this in Section 2.3.3,

The third limitation of PCA is that the identified dimensions may not be informative to discriminate data points of different classes. This is because the objective of PCA is to find dimensions that preserve the overall structure of data points, not to differentiate data points of different classes. For example, given images of printed uppercase letters O and Q, PCA might discover the gross features that characterize Os and Qs, but might ignore the tail that distinguishes an O and a Q. Linear discriminant analysis (LDA), which will be discussed in Section 2.4.1, addresses this shortcoming of PCA. It seeks the directions that are efficient to distinguish between letter O and letter Q.

2.1.2 Factor Analysis

Factor analysis [51] is another popular method for dimensionality reduction. Like PCA, factor analysis can be viewed as an attempt to approximate the covariance matrix. The essential purpose of factor analysis is to describe, if possible, the covariance relationships among multiple variables in terms of a few underlying, but unobservable, random quantities called *factors*. The factor model is motivated by the assumption that variables can be grouped by their correlations. That is, all variables within the same group are highly correlated among themselves but have relatively small correlations with variables in a different group. It is conceivable that each group of variables represents a single underlying construction, or factor, that is responsible for the observed correlations.

The Orthogonal Factor Model

Suppose the observable random vector $\mathbf{x}$, with m components/attributes, has mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. The factor model postulates that all the attributes in $\mathbf{x}$ are linearly independent given the unobservable random variables $y_1, \cdots, y_r$, called *common factors*, and m additional sources of variation $\epsilon_1, \cdots, \epsilon_m$, called *errors* or *specific factors*. In matrix notation, the factor model is expressed as:

$$\mathbf{x} = W\mathbf{y} + \boldsymbol{\mu} + \boldsymbol{\epsilon} \tag{2.4}$$

The $m \times r$ matrix W is call the *matrix of factor loadings*. The motivation is that, with $r < m$, the unobservable variables will offer a more parsimonious explanation of the dependencies between the observations. Furthermore, we introduce the following assumptions about the random vector $\mathbf{y}$ and $\boldsymbol{\epsilon}$,

$$\begin{aligned} &\mathrm{E}(\mathbf{y}) = \mathbf{0}, \quad \mathrm{Cov}(\mathbf{y}) = I \\ &\mathrm{E}(\boldsymbol{\epsilon}) = \mathbf{0}, \quad \mathrm{Cov}(\boldsymbol{\epsilon}) = \Psi = \mathrm{diag}(\psi_i, \cdots, \psi_m) \end{aligned} \tag{2.5}$$

and that $\mathbf{y}$ and $\boldsymbol{\epsilon}$ are independent,

$$\mathrm{Cov}(\mathbf{y}, \boldsymbol{\epsilon}) = \mathbf{0}_{r \times m}$$

These assumptions and the relation in (2.4) constitute the orthogonal factor model and the covariance structure for $\mathbf{x}$ can be inferred as:

$$\begin{aligned} \boldsymbol{\Sigma} &= \mathrm{Cov}(\mathbf{x}) \\ &= E\left(\mathbf{x} - \boldsymbol{\mu}\right)\left(\mathbf{x} - \boldsymbol{\mu}\right)^\top \\ &= E\left(W\mathbf{y} + \boldsymbol{\epsilon}\right)\left(W\mathbf{y} + \boldsymbol{\epsilon}\right)^\top \\ &= WW^\top + \Psi \end{aligned} \tag{2.6}$$

Apparently there exist an arbitrary rotation matrix R, such that W and WR generate the same covariance matrix $\boldsymbol{\Sigma}$.

The analysis of the factor model proceeds by imposing conditions that allow one to uniquely estimate $\mathbf{W}$ and $\mathbf{\Psi}$. The loading matrix is then rotated, where the rotation is determined by some 'easy-of-interpretation' criterion [71]. Once the loadings and the specific variances are obtained, factors are identified and estimated values for the factors themselves are constructed. Two popular methods of parameter estimation are the Principal Component method and Maximum Likelihood method[71].

Comparisons Between Factor Analysis and PCA

First, it's clear that a major distinction between factor analysis and PCA is that the factor analysis defines a specific generative model for the observed data patterns $\mathbf{x}$, while PCA does not.

Second, both factor analysis and PCA can be thought of as trying to preserve the covariance matrix $\mathbf{\Sigma}$ (or correlation matrix) as well as possible. The difference between them is that PCA concentrates on the diagonal elements, whereas in factor analysis the interest is in the off-diagonal elements [72].

To justify this statement, we first consider PCA. The objective is to maximize $\sum_{i=1}^{r} \text{var}(\mathbf{y}_i) = \sum_{i=1}^{r} \lambda_i$, which accounts for as much as possible the sum of diagonal elements of $\mathbf{\Sigma}$. Turning now to factor analysis, consider the factor model (2.4) and the corresponding equation (2.7) for $\mathbf{\Sigma}$. It can be seen that since $\mathbf{\Psi}$ is diagonal, the common factor term $W\mathbf{y}$ in (2.4) accounts completely for the off-diagonal elements of $\mathbf{\Sigma}$ in the perfect factor model, but there is no compulsion for the diagonal elements to be well explained by the common factors. The elements, $\psi_j, j = 1, \cdots, m$, of $\mathbf{\Psi}$ will be low if all of the variables have considerable common variation, but if a variable x_j is almost independent of all other variables, then $\psi_j = \text{var}(\epsilon_j)$ will be almost as large as $\text{var}(x_j)$. Thus, factor analysis concentrates on explaining only the off-diagonal elements of $\mathbf{\Sigma}$ by a small number of factors whereas conversely PCA concentrates on the diagonal elements of $\mathbf{\Sigma}$.

The third difference between the two techniques is the number of dimensions r that is

sufficient to represent the data in m dimensional space [72]. In PCA, if one variable is independent from other variables, it usually corresponds to a principal component. In PCA it is usually necessary to include such a principal component as part of the space with reduced dimensionality because they provide information that is not available from other variables. In contrast, in factor analysis, an independent attribute is usually accounted for by the noise model and therefore is not explained by the hidden factor. Thus, for most data sets the number of factors required by factor analysis is usually no larger – and may be strictly smaller – than the number of principal components required to account for the variation in the data.

One more difference between PCA and factor analysis is that it usually has more impact on factor analysis than it does on PCA [72] by varying the value of dimensionality of the model. In particular, in PCA when r is increased from r_1 to r_2, an additional $(r_2 - r_1)$ principal components are extracted and included to generate the subspace of the reduced dimensionality. In contrast, this is not true in factor analysis. When r is increased from r_1 to r_2, we often find that the common factors found by factor analysis could be completely different from the factors that are identified when $r = r_1$.

2.1.3 Probabilistic PCA

Probabilistic PCA (PPCA) was first introduced by Topping and Bishop [155, 154], where they showed how the principal axes of a set of observed data vectors may be determined through maximum likelihood estimation of parameters in a latent variable model closely related to factor analysis. An EM algorithm was also introduced [155, 154] to efficiently solve PPCA without having to compute the eigenvectors of a matrix. Although PPCA was originally designed to interpret PCA from a viewpoint of probabilistic model, it has several advantages when compared to the original PCA. First, the PPCA can be applied to estimate the principal axes even when some of the elements in the data matrix are missing [155, 131]. Second, PPCA provides a natural way to construct more complex projection methods, for

instance, by combining multiple PCA models [155].

The Probabilistic Model

In the case of isotropic noise $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \sigma^2 I)$, equation (2.4) implies the conditional probability distribution of $\mathbf{x}$ given $\mathbf{y}$ is expressed as:

$$\mathbf{x}|\mathbf{y} \sim N\left(W\mathbf{y} + \boldsymbol{\mu}, \sigma^2 I\right) \tag{2.7}$$

Assuming a Gaussian prior for the latent variable $\mathbf{y}$ defined by

$$\mathbf{y} \sim N\left(\mathbf{0}, I\right), \tag{2.8}$$

we can obtain the marginal distribution of $\mathbf{x}$ by integrating out the latent variables. The resulting distribution for $\mathbf{x}$ is again Gaussian, and is expressed as:

$$\mathbf{x} \sim N\left(\boldsymbol{\mu}, C\right), \tag{2.9}$$

where $C = WW^\top + \sigma^2 I$.

We now try to derive the conditional distribution of the hidden factor $\mathbf{y}$ given $\mathbf{x}$. Using Bayes' rule, the posterior distribution of the latent variables $\mathbf{y}$ given observed $\mathbf{x}$ is again Gaussian:

$$\mathbf{y}|\mathbf{x} \sim N\left(M^{-1}W^\top\left(\mathbf{x} - \boldsymbol{\mu}\right), \sigma^2 M^{-1}\right) \tag{2.10}$$

where $M = W^\top W + \sigma^2 I$, and note that M is of size $r \times r$ while C is $m \times m$

The log-likelihood of the observed data under this model is:

$$L = \sum_{i=1}^{n} \ln\left(p\left(\mathbf{x}_i\right)\right) = -\frac{n}{2}\left(d\ln\left(2\pi\right) + \ln|c| + tr\left(C^{-1}S\right)\right) \tag{2.11}$$

where

$$S = \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{x}_i - \boldsymbol{\mu}\right)\left(\mathbf{x}_i - \boldsymbol{\mu}\right)^\top$$

is the sample covariance matrix of the observation.

The Maximum-likelihood Estimators

Estimates for W and σ^2 may be obtained by EM algorithm [155] or may be obtained explicitly with C which is $WW^\top + \sigma^2 I$. Given σ, the noise level, the optimal solution to the loading matrix W that maximizes the log-likelihood L, is maximized when:

$$W_{ML} = U_r \left(\Lambda_r - \sigma^2 I\right)^{\frac{1}{2}} R \tag{2.12}$$

where the r column vectors in the $m \times r$ matrix U_r are the principal eigenvectors of S, with corresponding eigenvalues $\lambda_1, \cdots, \lambda_r$ in the $r \times r$ diagonal matrix Λ_r, and R is an arbitrary $r \times r$ orthogonal rotation matrix.

It may also be shown that, for $W = W_{ML}$, the maximum-likelihood estimation for σ^2 is given by:

$$\sigma^2_{ML} = \frac{1}{m-r} \sum_{j=r+1}^{m} \lambda_j \tag{2.13}$$

which is a clear interpretation as the variance 'lost' in the projection, averaged over the lost dimensions.

In practice, to find the most likely model given S, we would first estimate σ^2 from (2.13), and then W_{ML} from (2.12). For simplicity we would effectively ignore the rotation matrix R, i.e. choose $R = I$ [153]. Alternatively, an EM algorithm was also developed in [153] and closely related works can be found in [47, 131].

Zero Noise Limit

In conventional PCA, the reduced-dimensionality transformation of a data point $\mathbf{x}_i$ is given by $\mathbf{y}_i = U^\top (\mathbf{x}_i - \boldsymbol{\mu})$ and its reconstruction by $\hat{\mathbf{x}}_i = U\mathbf{y}_i + \boldsymbol{\mu}$. However, in the probabilistic framework of PPCA, the probabilistic analogue of the dimensionality reduction process of conventional PCA would be to invert the conditional distribution $p(\mathbf{x}|\mathbf{y})$ using Bayes' rule, to give $p(\mathbf{y}|\mathbf{x})$. In this case, each data point $\mathbf{x}_i$ is represented in the latent space not by a single vector, but by the Gaussian posterior distribution defined by (2.10). As an alternative to

the standard PCA projection, a convenient summary of this distribution and representation of $\mathbf{y}_i$ would be the posterior mean

$$\langle \mathbf{y}_i | \mathbf{x}_i \rangle = M^{-1} W_{ML}^{\top} (\mathbf{x}_i - \boldsymbol{\mu})$$

However, it is worth emphasizing that $W_{ML}\langle \mathbf{x}_i \rangle + \boldsymbol{\mu}$ is not the optimal linear reconstruction of $\mathbf{x}_i$(in the squared reconstruction-error sense). This may be seen from the fact that for $\sigma^2 > 0$, $M^{-1}W_{ML}^{\top}$ is not an orthogonal projection of $\mathbf{x}_i$. If we consider the limit as $\sigma^2 \to 0$, then $W_{ML} = U\Lambda^{\frac{1}{2}}$ (if we ignore the rotation matrix R) and the projection

$$W_{ML}\langle \mathbf{x}_n \rangle = W_{ML} \left(W_{ML}^{\top} W_{ML} \right)^{-1} W_{ML} (\mathbf{x}_i - \boldsymbol{\mu})$$

does become orthogonal and is equivalent to conventional PCA, but then the density model is singular and thus undefined.

2.1.4 Metric Multidimensional Scaling

Classical Multidimensional Scaling

Classical multidimensional scaling(MDS) [28] computes the low dimensional representations that most faithfully preserve the inner products between the high dimensional data points. It is usually cast into the following optimization problem:

$$\min_{\mathbf{y}} \quad \sum_{ij} (\mathbf{x}_i^{\top} \mathbf{x}_j - \mathbf{y}_i^{\top} \mathbf{y}_j)^2 = \|G - K\|_F \tag{2.14}$$

where$\|\cdot\|_F$ represents the Frobenius norm. G and K are the Gram matrix of the inputs and outputs respectively, with $G_{ij} = \mathbf{x}_i^{\top} \mathbf{x}_j$ and $K_{ij} = \mathbf{y}_i^{\top} \mathbf{y}_j$.

In many applications, the relationship between two data points is usually represented by their distance instead of their pair-wise similarity (i.e., dot product). Let $D_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^2$ and D be the matrix of squared pair-wise Euclidean distances. It is easy to derive matrix G from matrix D as:

$$G = -\frac{1}{2}(I - \frac{1}{n}\mathbf{1}\mathbf{1}^{\top}) D (I - \frac{1}{n}\mathbf{1}\mathbf{1}^{\top})$$

where **1** denotes the vector of all ones.

The solution to MDS is obtained from the eigen-decomposition of the Gram matrix G. Suppose $G = \sum_{k=1}^{n} \lambda_k \mathbf{v}_k \mathbf{v}_k^\top$, where λ_k is the k-th largest eigenvalue of G and $\mathbf{v}_k$ is the corresponding eigenvector. The r dimensional outputs of MDS are given by

$$\mathbf{y}_i \;\; = \left[\sqrt{\lambda_i}(\mathbf{v}_1)_i \cdots \sqrt{\lambda_r}(\mathbf{v}_r)_i\right]^\top, \quad i = 1, \cdots n$$

Relationships Between Classical scaling and PCA

Essentially, MDS and PCA produce the same outputs but in different formats. Recall that in PCA we write the covariance matrix as $C = XX^\top$ while in MDS the Gram matrix is $G = X^\top X$, where $X = [\mathbf{x}_1, \cdots, \mathbf{x}_n]$. The equivalence of their outputs can be easily established using the singular value decomposition which is:

$$\Lambda^{pca} = \Lambda^{mds}, \quad V^{pca} = XV^{mds}$$
$$Y^{pca} = (\Lambda^{pca})^{\frac{1}{2}} Y^{mds}$$

Here, V^{pca} and V^{mds} are eigen matrices containing eigenvectors of C and G as columns respectively; Λ^{pca} and Λ^{mds} are diagonal matrices whose diagonal elements are the corresponding eigenvalues of C and G.

2.1.5 Kernel PCA

In recent years, we have witnessed an explosion of work on *kernel methods* [138]. The basic idea of the kernel methods is to use the "kernel trick" that map a point $\mathbf{x}$ in the original r dimensional space $\mathcal{X}$ to a point $\mathbf{\Phi}(\mathbf{x})$ in a $\mathbf{N}_{\mathcal{F}}$-dimensional feature space $\mathcal{F}$, where $\mathbf{\Phi}(\mathbf{x}) = \left(\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \cdots, \phi_{\mathbf{N}_{\mathcal{F}}}(\mathbf{x})\right)^\top$. We can think of each function $\phi_i(\mathbf{x})$ as a non-linear mapping. The central idea of the kernel trick is based on the observation that for many algorithms, the key quantity is in the form $\mathbf{\Phi}(\mathbf{x}_i) \cdot \mathbf{\Phi}(\mathbf{x}_j)$. Hence, instead of explicitly defining the mapping functions $\Phi(\mathbf{x}_i)$, we only need to define the kernel function that assesses the dot product between $\Phi(\mathbf{x}_i)$ and $\Phi(\mathbf{x}_j)$, i.e., $k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{\Phi}(\mathbf{x}_i) \cdot \mathbf{\Phi}(\mathbf{x}_j)$. Kernel PCA [136, 137]

applies the kernel idea to perform PCA in the feature space $\mathcal{F}$. It is important to note that since the projection is performed in the space $\mathbf{N}_{\mathcal{F}}$ whose dimension can be significantly larger than m, the number of "reduced" dimensions can actually exceed m.

Performing PCA in Dot Product Space $\mathcal{F}$

Given data points $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$, we assume them to be in the vector space $\mathcal{X}$. Kernel PCA computes the principal components of the mapped points $\mathbf{\Phi}(\mathbf{x}_1), \mathbf{\Phi}(\mathbf{x}_2), \cdots, \mathbf{\Phi}(\mathbf{x}_n)$ in space $\mathcal{F}$. Since $\mathcal{F}$ may be infinite dimensional, the PCA problem needs to be transformed into a problem that can be solved in terms of the kernel $k(\cdot, \cdot)$. To this end, we consider the covariance matrix in $\mathcal{F}$, which is:

$$C = \frac{1}{n} \sum_{i=1}^{n} \left(\mathbf{\Phi}_i - \boldsymbol{\mu}\right) \left(\mathbf{\Phi}_i - \boldsymbol{\mu}\right)^{\top}$$

where $\mathbf{\Phi}_i = \mathbf{\Phi}(\mathbf{x}_i)$ and $\boldsymbol{\mu} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{\Phi}_i$. We are looking for eigenvector solutions $\mathbf{v}$ of

$$\mathbf{C}\mathbf{v} = \lambda \mathbf{v}.$$

Since this can be written as

$$\frac{1}{n} \sum_{i=1}^{n} \left(\mathbf{\Phi}_i - \boldsymbol{\mu}\right) \left[\left(\mathbf{\Phi}_i - \boldsymbol{\mu}\right) \cdot \mathbf{v}\right] = \lambda \mathbf{v}$$

the eigenvectors $\mathbf{v}$ lie in the span of the $\mathbf{\Phi}_i - \boldsymbol{\mu}$'s, e.g.,

$$\mathbf{v} = \sum_{i=1}^{n} \alpha_i \left(\mathbf{\Phi}_i - \boldsymbol{\mu}\right),$$

for some α_i. Thus, we can have m equations

$$\left(\mathbf{\Phi}_i - \boldsymbol{\mu}\right)^{\top} \mathbf{C}\mathbf{v} = \lambda \left(\mathbf{\Phi}_i - \boldsymbol{\mu}\right)^{\top} \mathbf{v}.$$

Now, consider the "kernel matrix" K_{ij}, the matrix of dot product in $\mathcal{F}$: $K_{ij} = \mathbf{\Phi}_i \cdot \mathbf{\Phi}_j, i, j = 1, \cdots, m$. Having the "centering matrix" $H = I - \frac{1}{n}\mathbf{1}\mathbf{1}^{\top}$, the above equation becomes

$$K^c K^c \boldsymbol{\alpha} = \bar{\lambda} K^c \boldsymbol{\alpha}$$

where $K^c = HKH$, $\boldsymbol{\alpha} \in \mathbb{R}^m$ and $\bar{\lambda} = m\lambda$.

It is also straightforward to show we only need to consider the following equation

$$K^c \boldsymbol{\alpha} = \bar{\lambda} \boldsymbol{\alpha}$$

Finally, to use the eigenvectors $\mathbf{v}$ to compute principal components in $\mathcal{F}$, we need $\mathbf{v}$ to be normalized, that is $\mathbf{v} \cdot \mathbf{v} = 1 = \bar{\lambda}\boldsymbol{\alpha} \cdot \boldsymbol{\alpha}$, so the $\boldsymbol{\alpha}$ must be normalized to have length $\frac{1}{\sqrt{\bar{\lambda}}}$.

Thus, for a given test point $\boldsymbol{\Phi}(\mathbf{x})$, the p-th eigenvector in feature space $\mathcal{F}$ is calculated by:

$$\langle \mathbf{v}^p, \boldsymbol{\Phi}(\mathbf{x}) \rangle = \frac{1}{\sqrt{\bar{\lambda}}} \sum_{i=1}^{n} \alpha_i^p \left(\boldsymbol{\Phi}\left(\mathbf{x}_i\right) \cdot \boldsymbol{\Phi}\left(\mathbf{x}\right) \right) = \frac{1}{\sqrt{\bar{\lambda}}} \sum_{i=1}^{n} \alpha_i^p k\left(\mathbf{x}_i, \mathbf{x}\right)$$

A Relationship Between Kernel PCA and Metric MDS

A kernel function is stationary if $k\left(\mathbf{x}_i, \mathbf{x}_j\right)$ depends only on the vector $\mathbf{x}_i - \mathbf{x}_j$. A stationary covariance function is isotropic if $k\left(\mathbf{x}_i, \mathbf{x}_j\right)$ depends only on the distance $d_{ij}^2 = \left(\mathbf{x}_i - \mathbf{x}_j\right) \cdot \left(\mathbf{x}_i - \mathbf{x}_j\right)$, so that we can write $k\left(\mathbf{x}_i, \mathbf{x}_j\right) = r\left(d_{ij}\right)$. Assume that the kernel matrix is scaled so that $r(0) = 1$. An example of an isotropic kernel is the RBF kernel $k\left(\mathbf{x}_i, \mathbf{x}_j\right) = e^{-\theta\left(\mathbf{x}_i - \mathbf{x}_j\right)^\top \left(\mathbf{x}_i - \mathbf{x}_j\right)}$, for some parameter $\theta > 0$.

Consider the Euclidean distance in feature space $\tilde{d}_{ij}^2 = \left(\boldsymbol{\Phi}_i - \boldsymbol{\Phi}_j\right)^\top \left(\boldsymbol{\Phi}_i - \boldsymbol{\Phi}_j\right)$. With an isotropic kernel, this can be re-expressed as $\tilde{d}_{ij}^2 = 2\left(1 - r\left(d_{ij}\right)\right)$. Thus, the matrix A has the elements $a_{ij} = r\left(d_{ij}\right) - 1$ which can be written as $A = K - \mathbf{1}\mathbf{1}^\top$. It can be easily verified that the centering matrix H annihilate $\mathbf{1}\mathbf{1}^\top$, so that $HAH = HKH$.

We see that the configuration of points derived from performing classical scaling on K aims to approximat the feature space distance computed as $\tilde{d}_{ij} = \sqrt{2\left(1 - r\left(d_{ij}\right)\right)}$.

If non-stationary kernels (for example, $k\left(\mathbf{x}_i, \mathbf{x}_j\right) = \left(1 + \mathbf{x}_i \cdot \mathbf{x}_j\right)^m$ with integer m) are used,we can again show the kernel MDS procedure operations on the matrix HKH. However, the distance $\tilde{d}_{ij}$ in the feature space is not a function of d_{ij}.

For a more detailed analysis, please refer to [165].

2.2 Graph-Based Method

Graph-based methods have recently emerged as a powerful tool for analyzing high dimensional data that are sampled from a low dimensional manifold and for revealing highly nonlinear structures [135]. These methods share the similar idea of using a spare graph (for simplicity, we assume that the graph is connected) for data representation, in which each node represents an input pattern and an edge represents a neighborhood relation to discretely approximate the submanifold sampled by the input patterns. From these graphs one can then construct matrices whose spectral decompositions reveal the low dimensional structure of the submanifold. In what follows, we review four widely used graph-based algorithms for manifold learning: Isomap [152], locally linear embedding [130, 134], Laplacian eigenmaps [5] and maximum variance unfolding [161, 150]

2.2.1 Isomap

The key idea behind Isomap [152] is to compute the low dimensional representation of a high dimensional data set that most faithfully preservers the pair-wise distances between input patterns as measured along the submanifolds from which they were sampled [135]. Isomap is built on the classical MDS but seeks to preserve the intrinsic geometry of the data which is captured in the geodesic manifold distances between all pairs of data points. The crux is estimating the geodesic distance between faraway points given only input-space distance. For neighboring points, input-space distance provides a good approximation to geodesic distance. For faraway points, geodesic distance can be approximated by adding up a sequence of "short hops" between neighboring points. These approximations are computed efficiently by finding the shortest path in the graph with edges connecting neighboring points.

The algorithm has three steps. The first step is to compute the k-nearest neighbors (k is pre-determined) of each input point and to construct a graph whose vertices represent input data points and whose edges connect k-nearest neighbors. In the second step, the edges are

assigned weights based on the Euclidean distances d_{ij} between all vertices along the shortest paths through the graph. One efficient algorithm to do this is Djikstra's method [35]. In the third step, the pair-wise distances d_{ij} from Djikstra's algorithm are fed as the input to classical MDS as described in Section 2.1.4, yielding low dimensional output $\mathbf{y}_i \in \mathbb{R}^r$ for which $\|\mathbf{y}_i - \mathbf{y}_j\|^2 \approx d_{ij}$. The value of r required for a faithful low dimensional representation can be estimated by the number of significant eigenvalues in the Gram matrix constructed by MDS.

Some theoretical analysis of Isomap can be found in [10, 39]. In [10] the authors proved that the two distance metrics, one along the submanifold and the other along the path of the graph, approximate each other arbitrarily close when the density of data points tends toward infinity. In [39] the authors further studied the conditions for the successful recovery of co-ordinates.

Several valuable extensions of Isomap include Conformal Isomap(C-Isomap) [33], Landmark Isomap(L-Isomap) [33], and the incremental Isomap [85] [84]. C-Isomap is capable of learning the structure of certain curved manifolds. This extension comes at the cost of making a uniform sampling assumption about the data. L-Isomap tries to address the two computational bottlenecks, the high cost in calculating the shortest-path distance matrix and the MSD eigen-decomposition, using a relatively small number of landmark points. The incremental Isoamp does not require all data points be available during training but gives us a way to update the low dimensional representation of data points gradually as more and more samples are collected. A similar extension that can handle out-of-sample data points can be found in [9]

2.2.2 Local Linear Embedding (LLE)

LLE [130, 134] is based on computing the low dimensional representation of a high dimensional data set that most faithfully preserves the local linear structure of nearby input patterns [135]. The algorithm differs significantly from Isomap in that its outputs are derived

from the smallest eigenvectors of a sparse matrix, as opposed to the largest eigenvectors of a (dense) Gram matrix.

The algorithm has three steps. The first step, as Isomap, is to compute the k-nearest neighbors of each high dimensional input point $\mathbf{x}_i$. In LLE however, we construct a directed graph which may or may not be symmetric. In this case, the set of edges E consists of ordered pairs (i, j) meaning that j is a neighbor of i, and we use $N_i = \{j|(i, i) \in E\}$ to denote the set of neighbors of i.

The second step of the algorithm assigns weights w_{ij} to the edges in this graph by solving the following least-squares problem:

$$\begin{aligned} \min_{w_{ij}} \quad & \sum_{i}^{n} \|\mathbf{x}_i - \sum_{j \in N_i} w_{ij}\mathbf{x}_j\|^2 \\ \text{s. t.} \quad & w_{ij} = 0, \text{ if } j \notin N_i \\ & \sum_{i \in N_i} w_{ij} = 1,\ i = 1, \cdots, n \end{aligned}$$

Here, LLE appeals to the intuition that each input pattern and its k-nearest neighbors can be viewed as samples from a small linear "patch" on a low dimensional submanifold and weights $w_{i,j}$ are computed by reconstructing each input point $\mathbf{x}_i$ from its k-nearest neighbors. The weights thus constitute a sparse matrix W which encodes local geometric properties of the data set by specifying the relation of each input pattern $\mathbf{x}_i$ to its k-nearest neighbors.

Note that the constrained weights that minimize these reconstruction errors obey an important symmetry: for any particular data point they are invariant to rotations, rescalings, and translations [134]. A consequence of these symmetries is that the reconstruction weights characterize geometric properties that do not depend on a particular frame of reference.

In the final step, LLE derives the output $\mathbf{y}_i \in \mathbb{R}^r$ that keeps the relations to its k-nearest neighbors which is another least-square problem:

$$\begin{aligned} \min_{\mathbf{y}_i} \quad & \sum_{i}^{n} \|\mathbf{y}_i - \sum_{j \in N_i} w_{ij}\mathbf{y}_i\|^2 \\ \text{s.t.} \quad & \sum_{i} \mathbf{y}_i = \mathbf{0} \\ & \frac{1}{n}\sum_{i} \mathbf{y}_i \mathbf{y}_i^\top = I \end{aligned} \tag{2.15}$$

It turns out that the solution to (2.15) can be obtained by computing the bottom $r+1$ eigenvectors of the matrix $(I-W)^\top(I-W)$. Let these normalized eigenvectors be $\mathbf{v}_n, \mathbf{v}_{n-1}, \cdots, \mathbf{v}_{n-r}$, associated with the bottom eigenvalues $0 = \lambda_n < \lambda_{n-1} \leq \cdots \leq \lambda_{n-r}$. We discard $\mathbf{v}_n = (\frac{1}{\sqrt{n}})\mathbf{1}$ associated with $\lambda_n = 0$, and use the next r eigenvectors to form the outputs as:

$$y_i = [(\mathbf{v}_{n-1})_i \cdots (\mathbf{v}_{n-r})_i]^\top, \; i = 1, \cdots, n$$

Free Parameters in LLE

In LLE, there are two free parameters: the number of nearest neighbors k and the target dimensionality r. The authors of [81, 80] suggested a hierarchical method for automatic selection of an optimal k. For r, [121] suggested to choose r by the number of eigenvalues comparable in magnitude to the smallest nonzero eigenvalue of the cost matrix $(I-W)^\top(I-W)$. However, the authors of [134] found that in practice it is not reliable and works only for contrived examples such as data lying on an essentially linear manifold, or data sampled in an especially uniform way so that the lowest nonzero eigenvalues are equal or nearly equal due to symmetry. Instead, they found it is more useful to rely on classical methods [118] for estimating the intrinsic dimensionality r of a data set. A similar idea can also be found in [127].

Extensions

Some useful extensions of LLE include LLE from pair-wise distances [134], convex reconstructions [134], out-of-sample extensions [9], and supervised LLE [32, 127, 128].

The LLE from pair-wise distance solves the problem when the user may not have access to data in the form of high dimensional vectors $\mathbf{x}_i$, but only the measurements of dissimilarity or distance between data points.

In the convex reconstructions, the author borrowed the idea from the non-negative matrix factorization [87] and additionally constrained the weights w_{ij} to be nonnegative, thus forcing the reconstruction of each data point to lie within the convex hull of its neighbors. Such a constraint has both advantages and disadvantages. On one hand, it tends to increase the robustness of linear fits to outliers. On the other hand, it can degrade the reconstruction of data points that lie on the boundary of a manifold and outside the convex hull of their neighbors. For such points negative weights may be helpful.

Supervised LLE [32, 127, 128] was introduced to deal with labeled data points. The intuition is that in order to obtain disjointed embeddings for individual classes, we would expect the local neighborhood of any sample $\mathbf{x}_i$ in class $y_i \in \{1, \ldots, c\}$ should be composed of samples belonging to the same class. This can be achieved by artificially increasing the pre-calculated distances between samples belonging to different classes, but leaving them unchanged if samples are from the same class, which is:

$$D' = D + \alpha \max(D)\Delta,\ \alpha \in [0, 1]$$

where $\Delta_{ij} = 1$ if the data point $\mathbf{x}_i$ and $\mathbf{x}_j$ are in the same class, otherwise it is 0. When $\alpha = 0$, one obtains unsupervised LLE; when $\alpha = 1$, the result is a fully supervised LLE. In fact, one can further show that this supervised version of LLE behaves as a nonlinear Fisher mapping (or linear discriminant analysis, LDA).

2.2.3 Laplacian Eigenmaps

Laplacian Eigenmaps [4, 5] is a nonlinear dimensionality reduction method. Similar to the other graph-based approach, LE constructs a neighborhood graph for given data that incorporates the pairwise relationship among the data. Using the notion of graph Laplacian, LE derives a low dimensional representation for a given data set that preserves the local neighborhood of individual data points. The low dimensional mapping generated by LE can also be viewed as a discrete approximation to a continuous mapping that naturally arises from the geometry of the manifold.

Alrgorithm

Given a data set $D = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$, where $\mathbf{x}_i \in \mathbb{R}^m, i = 1, \cdots n$, we first compute the k-nearest neighbors of each data point $\mathbf{x}_i$ and construct a symmetric undirected graph. In the graph, each node corresponds to an individual data point. Two nodes are connected by an edge if one node is a local neighbor of the other node. In the second step, we assign positive weights W_{ij} to the edges of the constructed graph. According to the LE algorithm, the weight is computed by a heat kernel that is defined as follows: if nodes i and j are connected by an edge in the graph, we compute the weight for the edge, denoted by W_{ij}, as

$$W_{ij} = \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma^2}\right),$$

otherwise, $W_{ij} = 0$ if node i and j are disconnected. In the last step of the algorithm, we derive the low dimensional representation of the data points by computing the eigenvalues and the eigenvectors of the following generalized eigenvector problem:

$$L\mathbf{v} = \lambda D\mathbf{v}$$

where D is a diagonal matrix with diagonal elements D_{ii} computed as $D_{ii} = \sum_j W_{ij}$. Matrix $L = D - W$ is often referred to as the Laplacian matrix, and is a well known quantity in the Graph theory for characterizing the properties of a weighted graph. Let

$0 = \lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_{n-1}$ be the eigenvalues of the graph Laplacian L and $\mathbf{v}_0, \cdots, \mathbf{v}_{n-1}$ be the corresponding eigenvectors. We construct the r dimensional embedding of the original data set by using the first $r+1$ eigenvectors of L, i.e., $\mathbf{v}_1, \ldots, \mathbf{v}_r$. In particular, the embedding vector of data point $\mathbf{x}_i$, denoted by $\mathbf{y}_i$, is computed as $y_i = ([v_1]_i, \ldots, [v_r]_i)$. Note that we did not use $\mathbf{v}_0$ in the construction of the low dimensional embedding since its corresponding eigenvalue is zero.

Optimal Embeddings

As illustrated in [5], we use the following example to demostrate why the embedding generated by the Laplacian Eigenmap algorithm preserves the local neighborhood structure.

Given a data set and its weighted graph $G = (V, E)$ constructed by the LE algorithm, we consider the problem of mapping the weighted graph G to a line (i.e., one dimensional embedding). We denote by $\mathbf{y} = (y_1, \ldots, y_n)^\top$ the image of this one dimensional mapping. The objective of this mapping is to ensure that the connected points in the graph G should stay as close to each other as possible after the mapping. We can formulate this criterion into the following optimization problem

$$\min_{\mathbf{y}} \sum_{ij} (y_i - y_j)^2 W_{ij}.$$

Specifically, the objective function with the choice of weights W_{ij} incurs a penalty if neighboring points are mapped far apart.

It turns out that for any $\mathbf{y}$, the objective $\frac{1}{2}\sum_{ij}(y_i - y_j)^2 W_{ij}$ can be further written as $\mathbf{y}^\top L\mathbf{y}$. The above problem thus can be expressed as:

$$\begin{aligned} \min_{\mathbf{y}} \quad & \mathbf{y}^\top L\mathbf{y} \\ \text{s. t.} \quad & \mathbf{y}^\top D\mathbf{y} = 1 \\ & \mathbf{y}^\top D\mathbf{1} = 0. \end{aligned}$$

Note that we introduce two constraints into the optimization problem. The first constraint $\mathbf{y}^\top D\mathbf{y} = 1$ is introduced to remove an arbitrary scaling factor in the embedding; the second

constraint $\mathbf{y}^\top D\mathbf{1} = 0$ is introduced to eliminate the trivial solution that collapses all vertices of G into a single number. We can also interpret the second constraint as the removal of a translation invariance in the mapping $\mathbf{y}$.

We can generalize the above discussion to the problem of embedding a graph into r-dimensional Euclidean space. Let $Y = (\mathbf{y}_1 \cdots \mathbf{y}_r)$ be the resulting mapping, where $\mathbf{y}^{(i)}$, i.e., the ith row of matrix Y, is the low dimensional representation of data point $\mathbf{x}_i$. Similar to the above discussion we cast it into the following optimization problem:

$$\begin{aligned} \min_{Y} \quad & \sum_{ij} \|\mathbf{y}^{(i)} - \mathbf{y}^{(j)}\|^2 w_{ij} = \mathrm{tr}(Y^\top L Y) \\ \text{s. t.} \quad & Y^\top D Y = I. \end{aligned}$$

The constraint $Y^\top DY = I$ ensures that the dimensionality of the resulting subspace is r. The solution of the above optimization problem is given by the first $r+1$ smallest eigenvectors of the generalized eigenvalue problem $L\mathbf{y} = \lambda D\mathbf{y}$.

Laplacian Eigenmap and LLE

Laplacian eiganmap and LLE are similar in that both methods try to preserve the local neighborhood structure and the low dimensional embeddings are derived by the smallest eigenvectors of certain matrices. In [5], the authors analyzed the connection between the two methods. Recall that we use the semi-definite matrix $(I - W)^\top (I - W)$ in LLE and the Laplacian matrix L in Lalacian eigenmap. Both matrices can be treated as operators acting in functions defined in the data points. In [5] the authors showed that under certain conditions, the generalized eigenvector problem of LLE can be expressed as $L^2 v = \lambda v$. Since L^2 and L share the same eigen spectrum, we have that LLE and LE share the same low dimensional embedding.

Extensions

Several extensions of Laplacian eigenmap have recently been developed. In [58], the author introduced an linear version of Laplacian eigenmap called Locality Preserving Projections (LPP), for which a linear projection $\mathbf{y}^\top = \mathbf{a}^\top X$ is preferred in the third step of the Laplacian eigenmap algorithm, where $\mathbf{a}$ is a transformation vector. As a result, the generalized eigenvector problem becomes $XLX^\top \mathbf{a} = \lambda XDX^\top \mathbf{a}$. The key advantage of LPP in comparison to LE is that it is computationally more efficient and can easily handle the out-of-sample problem.

In [4, 6], the authors extended the idea of Laplacian eigenmap to semi-supervised learning, to which they refered as manifold regularization. The central idea of manifold regularization is that classification functions are naturally defined only on the submanifold in question rather than the total ambient space. Using the Laplace-Beltrami operator, one produces a basis (the Laplacian Eigenmaps) for a Hilbert space of square integrable functions on the submanifold. To recover such a basis only unlabeled examples are required. Once such a basis is obtained, training can be performed using the labeled data set.

2.2.4 Maximum Variance Unfolding

Maximum variance unfolding(MVU)[161, 162, 135] is also known as semi-definite embedding(SDE) [159, 160, 163], which was first proposed by Weinberger and Saul in 2004. The algorithm tries to "unfold" the manifold by pulling the data points apart as far as possible, while faithfully preserving the local distances and angles between nearby input data points.

The MVU Algorithm

Similar to the other graph-based approaches, the first step of MVU is to compute the k-nearest neighbors of each input pattern. A neighborhood-indicator matrix η is defined as $\eta_{ij} = 1$ if and only if one input pattern is the k-nearest neighbor of another, or both input patterns are the k-nearest neighbors of some data point. In other words, we create a fully

connected clique of size $k+1$ out of every input $\mathbf{x}_i$ and its k nearest neighbors. In order to preserve both the distances and angles between the k-nearest neighbors, we introduce the following objective function to measure the quality of the embedding $\mathbf{y} = (y_1, \ldots, y_n)$

$$\|\mathbf{y}_i - \mathbf{y}_j\|^2 = \|\mathbf{x}_i - \mathbf{x}_j\|^2. \tag{2.16}$$

Note that although the objective in the above only aims to preserve the distance, it also preserves the angle between points in the k-nearest neighbors. This is because the graph is constructed based on the clique of the nearest neighbors. To eliminate a translational degree of freedom in the low dimensional representation, the outputs are also constrained to be centered on the origin leading to the following constraint

$$\sum_i \mathbf{y}_i = \mathbf{0}. \tag{2.17}$$

Finally, since the algorithm attempts to "unfold" the input patterns, it aims to maximize the variance of the mapping, i.e., $\sum_{i=1}^n \|\mathbf{y}_i\|_2^2$. By putting both the objective and the constraints together, we have the following optimization problem for MVU algorithm:

$$\begin{aligned} \max_Y \quad & \sum_i \|\mathbf{y}_i\|^2 = \frac{1}{2n} \sum_{ij} \|\mathbf{y}_i - \mathbf{y}_j\|^2 \\ \text{s. t.} \quad & \sum_{i=1}^n \mathbf{y}_i = \mathbf{0} \\ & \|\mathbf{y}_i - \mathbf{y}_j\|^2 = \|\mathbf{x}_i - \mathbf{x}_j\|^2, \text{ for } \eta_{ij} = 1 \end{aligned} \tag{2.18}$$

The optimization problem listed above is not convex due to the non-convex constraints. However, it can be converted into a convex optimization when using semi-definite embedding [156](see Appendix C), i.e.,

$$\begin{aligned} \max_K \quad & \operatorname{Tr} K \\ \text{s. t.} \quad & K \succeq 0, \quad \sum_{ij} K_{ij} = 0 \\ & K_{ii} + K_{jj} - 2K_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^2, \text{ for all } (i,j) \text{ with } \eta_{ij} = 1 \end{aligned} \tag{2.19}$$

where, $K_{ij} = \mathbf{y}_i \cdot \mathbf{y}_j$ denotes the Gram matrix of the output. After obtaining the Gram matrix K by solving the above optimization problem, similar to MDS, we derive the low dimensional embedding by computing the principle eigenvectors of the Gram matrix K.

Extensions

For real-world problems, it may be desirable to relax the constraints in (2.19) such that the local distances are preserved only to some approximation. One trick often used is to replace the strict equalities by inequalities [162]. More specifically, we use $K_{ii} + K_{jj} - 2K_{ij} \leq \|\mathbf{x}_i - \mathbf{x}_j\|^2$ for all (i, j) with $\eta_{ij} = 1$ to replace the corresponding equality constraint in (2.19). Interestingly, the dual of this relaxed SDP problem is equivalent to the problem studied by Sun and Xiao in the calculation of the fastest mixing Markov processes on a weighted graph [150, 167]. Additional relaxations of the SDP problem in (2.19) can be found in [162, 14].

One major concern with the MVU algorithm is that in practice it scales poorly to large data sets because it requires solving an SDP problem over a matrix of size $n \times n$, where n is the number of examples. In order to address this problem, [159] suggested to approximate the Gram matrix K by $K = QLQ^\top$, where Q is a $n \times k$ matrix that transforms Gram matrix K of size $n \times n$ into a landmark matrix L of size $k \times k$. This approximation allows us to solve an SDP problem of significantly smaller size, making the MVU scale well even to large data sets. The transformation matrix Q can be derived by solving a sparse set of linear equations as suggested in [159]. A similar approach is proposed in [164] except that factorized matrix Q is derived by expanding the solution of the original problem in terms of the bottom eigenvectors of a graph Laplacian.

Connecting to the Other Methods

It is evident that the objective function of MVU is closely connected to PCA in that both methods are related to the maximization of data variance. MVU is also closely related to MDS since both methods aim to keep the pair-wise distance between nearby data points.

Isomap can be interpreted as an approximation to the MVU problem. Since the Euclidean distance between any two points on a manifold is always smaller than their geodesic distance, the Euclidean distance provides an upper bound for the corresponding geodesic distance. In addition, maximizing the variance is equivalent to maximizing the total pair-wise Euclidean distance in the embedding space. Therefore, in this sense Isomap attempts to maximize the variance by directly using the geodesic distances. This interpretation becomes accurate in the limit with increasing sampling density ($n \to \infty$), if the sub-manifold is isometric to a convex subset of the Euclidean space. In particular, this condition guarantees the asymptotic convergence of Isomap algorithm [10, 39]. In this case, the pair-wise geodesic distances become feasible to the MVU problem, and the solution to MVU approaches its upper bound obtained by Isomap, thus MVU converges to the same limit as Isomap. If the above condition is not satisfied, then Isomap and MVU could behave quite differently [160]. More general conditions for the asymptotic convergence of MVU remains an open question.

For the connections between MVU LLE and Laplacian eigenmaps, [167] gives detailed discussions in which the authors showed the dual problem of MVU is closely connected to LLE and Laplacian eigenmaps.

2.2.5 From the Kernel Point of View

All of the algorithms presented in this section can be viewed as instances of kernel PCA [56, 135], with the kernel matrices that are derived from sparse weighted graphs rather than a pre-defined kernel function. Often, these kernels are described as data-dependent kernels because they are derived from graphs that encode the neighborhood relations of the input patterns in the training set.

The foundation of Isomap algorithm is MDS. As pointed out in Section 2.1.5, the Gram matrix constructed in MDS from these geodesic distances can be viewed as a kernel matrix so as to Isomap. For finite data sets however, this matrix in Isomap is not guaranteed to be positive semi-definite. It should therefore be projected onto the cone of positive semi-definite

matrices before it is used as a kernel matrix in other settings.

The graph Laplacian in the Laplacian eigenmap arises in the description of diffusion on the graph and can be related to Green's function and heat kernels [78]. The smallest eigenvectors of Laplacian L are equal to the top eigenvectors of the pseudo-inverse of the Laplacian, $L^\dagger$, which can further be viewed as a (centered) kernel matrix for kernel PCA. Moreover, viewing the elements $L^\dagger_{ij}$ as inner products, the squared distances defined by $L^\dagger_{ii} + L^\dagger_{jj} - L^\dagger_{ij} - L^\dagger_{ji}$ are in fact proportional to the round-trip commute times of the continuous-time Markov chain with transition rate matrix L. The commute times are nonnegative, symmetric, and satisfy the triangle inequality thus, Laplacian eigenmaps can be alternately viewed as MDS on the metric induced by these graph commute times.

The matrix diagonalized by LLE can also be interpreted as an operator on graphs whose pseudo-inverse corresponds to a kernel matrix. The operator does not generate a simple diffusive process, but in certain cases it acts similarly to the square of the graph Laplacian [56].

For MVU, if we treat the Gram matrix K in MVU as a kernel matrix, then the whole procedure of MVU can be thought as applying the kernel PCA on the learned kernel K. However, the kernel matrix K in MVU is interested in a very different aspect which is, in kernel PCA, the input data are typically mapped from the low dimensional space to a much higher dimensional feature space, while in MVU it is opposite, the kernel matrix K is used to map the inputs into a lower dimensional space. Another useful point is that in MVU, the kernel is learned without supervised information.

Summary

In this section we presented several graph-based algorithms for dimensionality reduction. Comparing to the linear methods, such as PCA and MDS presented in the previous section, the graph-based methods belong to the category of non-linear dimensionality reduction. Furthermore, Isomap can be viewed as a "global" method since it attempts to preserve all

geodesic distance; by contrast, LLE, Laplacian Eigenmap and MVU are "local" methods since only local neighbor structures are preserved through the mapping. Although these methods start from quite different geometrical consideration, they all look similar under the hood and some strong connections have been found among those methods.

In terms of scalability, LLE and Laplacian eigenmap are efficient and can scale to modestly large data sets (i.e., $n < 10000$), provided that one uses special-purpose eigensolvers that are optimized for sparse matrices. MVU is the most inefficient among all the graph-based approaches due to the expense of solving semi-definite programs over $n \times n$ matrices.

In summary, research on graph-based methods for dimensionality reduction continues at a rapid pace. Motivation for ongoing work include the handling of manifolds with more complex geometries, the need for robustness to noise and outliers, and the scalability to large data sets.

2.3 Matrix Factorization

2.3.1 Low Rank Approximation

The task of dimensionality reduction can also be viewed naturally from the point of matrix factorization. Consider the data set is organized in the observed matrix $X \in R^{m \times n}$, where the columns of X are data vectors $\mathbf{x}_i$. Our goal is to approximate X by a product of two matrices $UV^\top$, where $U \in R^{m \times r}$ and $V \in R^{n \times r}$. For each data vector $\mathbf{x}_i$, we have $\mathbf{x}_i \sim U\mathbf{v}_i^\top$, where $\mathbf{v}_i$ is the ith column of matrix V, implying that each data vector is approximated by a linear combination of column vectors in matrix U and the combination coefficients are given by matrix V. Geometrically speaking, each data vector $\mathbf{x}_i \in R^m$ is approximated by its projection in a r-dimensional subspace that is spanned by column vectors of U. For the convenience of discussion we denote $Y = UV^\top$ as the approximate matrix. Evidently the rank of Y is at most r.

The most common, and in many ways the simplest, method to measure how well the model Y "approximate" the data X, is the sum squared error or Frobenius distance between X and Y:

$$\|X - Y\|_F^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} (X_{ij} - Y_{ij})^2$$

Minimizing the Frobenius distance can also be seen as the maximum likelihood estimation in the presence of the additive i.i.d. Gaussian noise with a fixed variance in terms of a probabilistic model. In particular, we assume that the observed matrix X is generated by the following stochastic process

$$X = Y + Z,$$

where Y is a rank r matrix and each element in Z is independent and follows a Gaussian distribution with zero mean and constant variance σ^2. Then, the log-likelihood of Y given

the observed matrix X is

$$\begin{aligned}\log \Pr(Y|X) &= -\frac{mn}{2}\ln(2\pi\sigma^2) - \sum_{ij}\frac{(X_{ij}-Y_{ij})^2}{2\sigma^2} \\ &= -\frac{1}{2\sigma^2}\|X-Y\|_F + \text{const}\end{aligned}$$

As indicated by the above equation, finding Y by maximizing the likelihood is equivalent to minimizing the Frobenius distance between X and Y. It is worth pointing out that besides Frobenius distance, there are other approaches for measuring the discrepancy between data matrix X and the underlying model Y, such as matrix Bregman divergence [2]. More information can be found in [148].

In some applications, besides the rank constraint, additional constraints of factorized matrices are introduced to ensure the appropriateness of the resulting embedding. The most well known one is non-negative matrix factorization [87], which will be introduced in Section 2.3.3

2.3.2 Singular Value Decomposition

Singular value decomposition (SVD) is one of the most widely used low rank approximation techniques in dimensionality reduction. PCA, introduced in the Section 2.1, can be viewed as a special case of SVD.

Let X be an $m \times n$ matrix with rank k. We can write X as

$$X = U\Sigma V^\top,$$

where

- $\Sigma = \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix}$ is a matrix of size $m \times n$, where $D = \text{diag}(\sigma_1, \ldots, \sigma_k)$ is a diagonal matrix whose diagonal entries are the first k singular values of X;

- U is an orthogonal matrix of size $m \times m$, and V is an orthogonal matrix of size $n \times n$. The columns of U are eigenvectors of $XX^\top$ and are called left singular vectors of X, the columns of V are eigenvectors of $X^\top X$ and are called right singular vectors of X.

Note that since only D (i.e., a matrix of size $k \times k$) in Σ is nonzero, it is sufficient to compute the first k column vectors of matrices U and V in order to reconstruct the original data matrix X, i.e.,

$$X = U_{m\times k}\Sigma_{k\times k}V^{\top}_{k\times n} = U_{m\times k}DV^{\top}_{k\times n}$$

In the context of dimensionality reduction, we often need to solve the problem of approximating a matrix X of rank k with another matrix Y which has a lower rank r(i.e., $r < k$). For example, when the approximation is based on the minimization of the Frobenius norm of the difference between X and Y under the constraint that the rank of Y is r, the solution is given by the SVD of X, i.e., $Y = U_{m\times r}\text{diag}(\sigma_1, \ldots, \sigma_r)V^{\top}_{n\times r}$.

2.3.3 Non-negative Matrix Factorization

Directly applying SVD will result in factorized matrices that have both positive and negative elements which often contradict the physical meaning of the result. For example, if the data matrix X is used to represent the intensities of pixels in a gray-scale image, it is difficult to interpret the reconstructed matrix Y for a gray-scale image that has negative elements. In this case it is more appealing to find the reduced rank nonnegative factors to approximate a given nonnegative data matrix. This approach is often referred to as nonnegative matrix factorization (NMF) [87].

Definitions

Given a $m \times n$ data matrix X with $X_{ij} \geq 0$ and a pre-determined positive integer $r \ll \min(m, n)$, NMF finds two non-negative matrices $U \in R^{m\times r}$ and $V \in R^{n\times r}$, such that

$$X \approx UV^{\top}$$

A common way to find U and V is to minimize the Euclidean distance between X and $UV^{\top}$:

$$\begin{aligned} \min_{U,V} \quad & f(U,V) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(X_{ij} - (UV^{\top})_{ij}\right)^2 = \frac{1}{2}\|X - UV^{\top}\|_F^2 \\ \text{s. t.} \quad & U_{ia} \geq 0, V_{jb} \geq 0, \forall i, a, b, j. \end{aligned} \tag{2.20}$$

Besides the Frobenius norm, [88] provided an information theoretic formulation based on the Kullback-Leibler divergence of X from $UV^\top$, leading to the following optimization problem:

$$\begin{aligned} \min_{U,V} \quad & f(U,V) = \sum_{i=1}^{m}\sum_{j=1}^{n}\left(X_{ij}\log\frac{X_{ij}}{(UV^\top)_{ij}} - X_{ij} + (UV^\top)_{ij}\right) \\ \text{s. t.} \quad & U_{ia} \geq 0, V_{jb} \geq 0, \forall i,a,b,j. \end{aligned} \tag{2.21}$$

Other alternative objective functions that measure the divergence between X and its reconstruction $UV^\top$ can be found in [27, 157, 54, 34]. It is worth pointing out that in [37], the non-negative constraints of U and V are relaxed by only requiring V to be non-negative extending the application of NMF methods.

Algorithms

In most cases NMF algorithms can be classified into three general categories[11]: multiplicative update algorithms, gradient descent algorithms, and alternating least squares algorithms. [26] recently created a library (NMFLAB) of MATLAB routines that include most of the existing algorithms for NMF algorithms. Below we briefly review algorithms of these three categories.

Multiplicative update algorithms The Multiplicative update algorithm was first presented in [87]. It employs the following equations for updating the solution for U and V

$$V_{bj}^\top \leftarrow V_{bj}^\top \frac{\left(U^\top X\right)_{bj}}{\left(U^\top UV^\top\right)_{bj}} \qquad U_{ia} \leftarrow U_{ia}\frac{(XV)_{ia}}{(UV^\top V)_{ia}} \tag{2.22}$$

Usually, a very small number such as 10^{-9} is added to both of the denominators in (2.22) to avoid division by zero. In [88], Lee and Seung claimed that the above algorithm converges to a local minimum which was later shown to be incorrect[91, 43, 50, 24]! To understand why the updating equation in (2.22) may not lead to local optimal, we consider the Karush-

Kuhn-Tucker (KKT) conditions for solution U and V:

$$\begin{aligned} U &\geq 0 \\ V &\geq 0 \\ (UV^\top - X)V &\geq 0 \\ U^\top(UV^\top - X) &\geq 0 \\ (UV^\top - X)V \bullet U &= 0 \\ U^\top(UV^\top - X) \bullet V^\top &= 0, \end{aligned}$$

where $\bullet$ represents the element-wise product between two matrices. It's easy to show that if the optimal point(U^*, V^*) does not have any elements equal to 0, then it converges to a local minimum. However, if certain elements in U^* or V^* become zero after several iterations, for instance if V^*_{ij} is zero, one has to show $\frac{\partial f}{\partial V}(U^*, V^*) \geq 0$ in order to ensure the local optimality of the solution. It is not evident that the multiplicative update rules in (2.22) are able to deliver this guarantee. Because of this issue, the local optimality of the solution found by the multiplicative method is not always guaranteed. In [90], the authors propose an improved version of the Lee and Seung algorithm that is guaranteed to converge to a stationary point, although it requires slightly more work per iteration than the original Lee and Seung algorithm.

Despite the convergency problem, the Lee and Seung multiplicative update algorithm has become a common approach for solving NMF. Another shortcoming of the Lee and Seung algorithm is that it is notoriously slow to converge. Several modification have been proposed to address this problem [50, 34, 90].

Gradient descent algorithms The Gradient descent algorithms use the following equations for updating U and V

$$V^\top_{bj} \leftarrow V^\top_{bj} - \epsilon_V \frac{\partial f}{\partial V^\top_{bj}} \qquad U_{ia} \leftarrow U_{ia} - \epsilon_U \frac{\partial f}{\partial U_{ia}} \tag{2.23}$$

where ϵ_U and ϵ_V are the step sizes. In fact, the multiplicative update algorithms can also be considered as a gradient based method [24, 88]. The trick to making these algorithms efficient

comes with choosing appropriate values for the step sizes ϵ_U and ϵ_V. Various algorithms have been proposed that use different strategies for choosing the step size parameters ϵ_U and ϵ_V. For instance, some algorithms initially set these step size values to 1, then multiply them by one-half at each subsequent iteration [63]. Despite the simplicity, these algorithms suffer from the problem that they do not guarantee the elements of the updated matrices U and V to be non-negative. A common practice employed by many gradient descent algorithms is a projection step that projects the obtained solution to the non-negative orthant [140, 63, 24, 116]. Without a careful choice of ϵ_U and ϵ_V, it is difficult to analyze the convergence of the gradient descent methods.

Alternating least squares algorithms As suggested by the name, this method solves the optimization problem in (2.20) by optimizing variables alternately. Particularly, in each iteration of the algorithm, with one of the two variables (i.e., U and V) fixed, we optimize the other variable by solving a least square problem. Alternating least square algorithms were first used in [114]. This algorithm exploits the fact that while the optimization problem in (2.20) is not convex in both U and V, it is however convex in either U or V. Thus, given one matrix fixed, the other matrix can be found efficiently by a simple least square computation. The basic idea of alternating least squares algorithms is summarized as follows.

- Initialize U as a random dense matrix
- for $i = 1$ to max iter
    1. Solve for $V^\top$ in matrix equation $U^\top U V^\top = U^\top X$
    2. Set all negative elements in $V^\top$ to 0
    3. Solve for U in matrix equation $V^\top V U^\top = V^\top X^\top$
    4. Set all negative elements in U to 0

In the above procedure, the nonnegativity is ensured by using the simplest projection step which sets all negative elements resulting from the least squares computation to 0.

This simple projection step often results in a sparse solution for U and V. Moreover, it allows additional flexibility during the iteration that is not available in other algorithms as pointed out in [11]. For instance, one drawback of the multiplicative algorithms is that once an element in U or V becomes 0, it will remain as 0 forever. This locking of 0 elements is restrictive, meaning that once the algorithm starts heading down toward a fixed point, even if it is a poor fixed point, it must continue in that vein. The alternating least squares algorithms are more flexible since it allows the iterative process to escape from a poor path. Another appealing feature of alternating least squares algorithm is if the algorithm is implemented appropriately, they can be much faster than the other two types of algorithms. A discussion of the convergency of the alternating least squares algorithms can be found in [11].

2.3.4 Maximum Margin Matrix Factorization

The idea of Maximum Margin Matrix Factorization (MMMF) [148, 149, 125] arises from the study of collaborative filtering [17], in which the goal is to complete the missing elements in the user-item rating matrix. Unlike the matrix factorization methods discussed in the previous sections where the focus is to approximate an existing data matrix by a matrix with a lower rank, MMMF searches for the optimal factorized matrices that minimizes the trace norm of the approximate matrix. More specifically, given the factorization $X = UV^\top$, MMMF aims to find matrices U and V with the least norms. The idea of minimizing the norms of the factorized matrices arises by viewing matrix factorization as a problem of feature extraction. Unlike the formulation of matrix factorization presented in the previous sections that often result in a non-convex optimization problem, MMMF can be cast into a convex optimization, making it computationally more desirable.

Collaborative Filtering and Matrix Factorization

The objective of collaborative filtering is to predict the utility of items for a given user based on a database of user votes from a sample or population of other users [17]. Collaborative

filtering can be formalized as a matrix completion problem: Let Y be the user-item rating matrix, with each row corresponding to an user, each column corresponding to an item, and every element $Y_{i,j}$ representing the rating of user i for item j. Since each user only provides ratings for a subset of items, or in other words, only a part of the matrix is observed, the objective of predicting users' ratings for the remaining items can be naturally viewed as the completion of the matrix Y.

It is straightforward to extend the idea of low-rank matrix factorization for collaborative prediction [62, 100]. Given a partially observed matrix Y, we aim to find the matrices (U, V) that minimizes the discrepancy between the observed entries Y_S and the corresponding entries in $X = UV^\top$. With the estimated U and V, we can reconstruct the entire matrix Y, leading to the prediction of missing ratings in Y.

Matrix Factorization and Linear Prediction

In order to motivate the formulation for maximum margin matrix factorization, below we try to link the problem of matrix factorization and linear prediction. Suppose one of the factor matrices, say U, is fixed and only the other factor matrix $V^\top$ needs to be learned. Then, fitting each column of the target matrix Y by $UV^\top$, is a separate linear prediction problem; e.g., each row of U is a "feature vector"; and each column of $V^\top$ is a linear predictor that predicts the entries in the corresponding column of Y based on the "features" in U. According to the maximum margin principle [18], the optimal V is found by minimizing the reconstruction error as well as the square of Frobenius norm of U. It is the similarity between matrix factorization and linear prediction that leads to the maximum margin principle for matrix factorization.

In collaborative prediction, both U and V are unknown and need to be estimated. This can be thought of learning both the feature vectors (i.e., U) and the linear predictor (i.e., V) for Y simultaneously. Although a natural approach is to solve this problem by some alternating method, below we will review the maximum margin matrix factorization method

that allows us to determine both U and V at the same time.

Maximum-margin Low Trace Norm Matrix Factorization

The trace norm $\|X\|_{tr}$ is defined as the sum of the singular values of X [148] (See Appendix D for details), which can also be written as:

$$\|X\|_{tr} \quad = \min_{X=UV^\top} \|U\|_F \|V\|_F \quad = \min_{X=UV^\top} \frac{1}{2}\left(\|U\|_F^2 + \|V\|_F^2\right)$$

It is important to note that the trace norm $\|X\|_{tr}$ is a convex function of X. This fact is the key for casting the maximum margin matrix factorization problem into a convex optimization problem.

To simplify presentation, we focus on the problem with binary labels, i.e., every observed elements in Y is either $+1$ or -1. In the hard-margin matrix factorization framework, we seek the completed matrix X that (1) provides a good approximation for all the observed elements in Y, and (2) has the least trace norm. We can cast the above requirements into the following optimization problem:

$$\begin{aligned} \min_X \quad & \|X\|_{tr} \\ \text{s. t.} \quad & Y_{ia}X_{ia} \leq 1, \text{ for all } ia \in S \end{aligned} \tag{2.24}$$

where S includes the indices of all the observed elements in Y. Similarly, the soft-margin matrix factorization, where we minimize a trade-off between the trace norm and its hinge-loss relative to Y_S, is expressed as follows:

$$\min_X \quad \|X\|_{tr} + c \sum_{ia \in S} \max(0, 1 - Y_{ia}X_{ia}) \tag{2.25}$$

It is worth emphasizing that there is an inverse dependence between the norm and the margin: fixing the margin and minimizing the trace norm is equivalent to fixing the trace norm and maximizing the margin.

The geometric interpretation of the maximum matrix factorization can be seen clearly from constraining all rows of U and V to have small L_2 norm. More clearly, instead of

constraining the norm of rows in U and V on average, we replace the trace norm with

$$\|X\|_{max} = \min_{X=UV^\top} \left(\max_i \|U_i\|_2 \right) \left(\max_a \|V_a\|_2 \right)$$

where U_i and V_a are row vectors of U and V. Then the hard-margin low-max-norm prediction corresponds to mapping the users and items to points and hyper planes in a high-dimensional unit sphere such that each users hyperplane separates his positive and negative items with a large-margin (the margin being the inverse of the max norm).

The soft-margin matrix factorization (2.25) can be further formulated as a semi-definite optimization problem [148, 156] as:

$$\begin{aligned} \min \quad & \frac{1}{2}(\mathrm{tr}A + \mathrm{tr}B) + c \sum_{ia \in S} \xi_{ia} \\ \text{s. t.} \quad & \begin{pmatrix} A & X \\ X^\top & B \end{pmatrix} \geq 0, \quad Y_{ia} X_{ia} \geq 1 - \xi_{ia}, \quad \xi_{ia} \geq 0, \quad \forall ia \in S \end{aligned} \tag{2.26}$$

By introducing a dual variable Q_{ia} for each constraint on X_{ia}, the dual of (2.26)[148]is given as:

$$\begin{aligned} \max \quad & \sum_{ia \in S} Q_{ia} \\ \text{s. t.} \quad & \begin{pmatrix} I & (-Q \otimes Y) \\ (-Q \otimes Y)^\top & I \end{pmatrix} \geq 0, \quad 0 \leq Q_{ia} \leq c \end{aligned} \tag{2.27}$$

where $Q \otimes Y$ denotes the sparse matrix with $(Q \otimes Y)_{ia} = Q_{ia} Y_{ia}$ for $ia \in S$ and zeros elsewhere. Since the prime problem is strictly feasible, the duality gap between the prime problem and the dual problem is zero. Hence, both the prime problem and the dual formulation give the same solution. Note that unlike typical prime and dual problems, in SDP, recovering the optimal solution to the prime problem directly from a dual optimal solution is in fact non-trivial. However, at least for the hard-margin problem this is possible to derive the solution for the prime problem from the solution to the dual problem by exploring the complementary slackness conditions[148].

2.4 Supervised and Semi-Supervised Dimensionality Reduction

All the dimensionality reduction methods discussed in the previous sections are unsupervised, i.e., no supervision information is provided other than the input patterns of data points. The methods reviewed in this section belong to the category of supervised and semi-supervised dimensionality reduction. These algorithms are inspired by the fact that in many cases additional information, such as class labels and pairwise constraints, is provided besides the input patterns of data. The objective of supervised and semi-supervised dimensionality reduction is to exploit the supervisory information in order to either guide or regulate the search of the space with reduced dimensions. In this section, we will review several the most representative methods to illustrate the general idea of the supervised and semi-supervised dimensionality reduction. It is worth pointing out that the methods reviewed in this section have a very close relationship to the methods developed in this thesis, and therefore they serve as a good starting point for the remaining chapters. Finally, it is also worth emphasizing, the supervised and semi-supervised dimensionality reduction is closely related to distance metric learning and kernel learning. We refer the audience to [169, 177] for more information.

2.4.1 Linear Discriminate Analysis

As discussed in Section 2.1.1, although PCA finds components that are representative for a given dataset, these components however may not be useful for distinguishing data from different classes. Unlike PCA, Linear Discriminate Analysis (LDA) [44] aims to identify the low dimensional space that has the maximum class discrimination. It achieves this goal by projecting data onto a low dimensional space in which the separation of data points from different classes is maximized while the dispersion of data from the same class is simultaneously minimized.

Assume the data set D is grouped as:

$$D = \{X_1, X_2, \ldots, X_k\}$$

where k is the number of classes and $X_i \in \mathbb{R}^{m \times n_i}$ is the data matrix of the ith class. n_i is the number of data points of the ith class and $\sum_{i=1}^{k} n_i = n$.

In LDA three matrices are defined as follows which are: *within-class*, *between-class*, and *total* scatter matrices:

$$S_w = \frac{1}{n} \sum_{i=1}^{k} \sum_{\mathbf{x} \in X_i} (\mathbf{x} - \mathbf{c}_i)(\mathbf{x} - \mathbf{c}_i)^\top \tag{2.28}$$

$$S_b = \frac{1}{n} \sum_{i=1}^{k} n_i (\mathbf{c}_i - \mathbf{c})(\mathbf{c}_i - \mathbf{c})^\top \tag{2.29}$$

$$S_t = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{c})(\mathbf{x}_i - \mathbf{c})^\top \tag{2.30}$$

where $\mathbf{c}_i$ is the sample mean of the ith class and $\mathbf{c}$ is the mean of the whole data set. It can be shown that the three matrices satisfy the following relationship:

$$S_t = S_w + S_b$$

It is evident that the trace of S_w, denoted by $\mathrm{tr}(S_w)$, measures the overall distance between each data point and its class center (mean). Similarly, the trace of S_b, denoted by $\mathrm{tr}(S_b)$, measures the separation between each class center and the center of all data points. In the low dimensional space where each data point is transformed by a projection matrix $G \in \mathbb{R}^{m \times (k-1)}$, the three scatter matrices become:

$$S_w^L = G^\top S_w G, \;\; S_b^L = G^\top S_b G, \;\; S_t^L = G^\top S_t G$$

In order to ensure that in the space of reduced dimensions (a) all the classes are well separated and (b) the data points in the same class are close to each other, LDA cast the dimensionality reduction problem into the following optimization problem:

$$\begin{aligned} \max_{G} \quad & \frac{\mathrm{tr}\left(G^\top S_b G\right)}{\mathrm{tr}\left(G^\top S_w G\right)} \\ \text{s.t.} \quad & G^\top G = I \end{aligned} \tag{2.31}$$

The solution of LDA comes from the result of the generalized eigenvalue problem:

$$S_w^{-1} S_b \mathbf{v}_i = \lambda_i \mathbf{v}_i \tag{2.32}$$

and the eigenvectors corresponding to the $k-1$ largest eigenvalues form the columns of the projection matrix G.

Extensions of LDA One well know problem in LDA is the so called *singularity* or *under-sampled* problem. It arises when the number of data points is smaller than the dimensionality, leading to singular scatter matrices singular and consequentially a ill defined eigenvalue problem in (2.32).

Many attempts have been made to deal with the singularity problem. One category of approaches is to reduce the dimensionality before LDA is applied. PCA+LDA [3, 173, 101] is a representative approach in this category. It first applies PCA to the given data set to identify a space of p dimension in which all the scatter matrices are nonsingular. It then applied the standard LDA to data patterns in the reduced space. The parameter p is usually estimated by cross validation. Another popular method to deal with the singularity of scatter matrices is regularization, which is known as RLDA [55]. In RLDA, a small positive value is added to the diagonal elements of the scatter matrices, which makes them to be positive definite thus nonsingular [49]. Other methods, such as null space LDA (NLDA) [23], orthogonal LDA (OLDA) [171], and uncorrelated LDA (ULDA) [171], have also been proposed and applied successfully to resolve the singularity issue in different domains. An unified framework is presented in [69] that unifies these variants of LDA methods.

2.4.2 Semi-Supervised LLE and ISOMAP

Semi-supervised LLE [170] and ISOMAP [170] can be viewed as a natural extension of their unsupervised versions which were reviewed in the Section 2.2. In semi-supervised settings, the prior information of the exact mapping of certain data points is known and is used to guide the procedure of dimensionality reduction. Without loss of generality, we assume that

for n_l data points, we know their low dimensional representation. By rearranging the data matrix D, we can partition D into two parts as $D = \{X_l, X_u\}$, where X_l represents the data points for which we know their low dimensional representation and X_u are other data points. Accordingly, we can partition the low dimensional representation Y into two parts as $Y = \{Y_l, Y_u\}$, where Y_l is given and Y_u needs to be computed.

Semi-Supervised LLE From (2.15), the objective of LLE is to minimize $\sum_i^n \|\mathbf{y}_i - \sum_{j \in N_i} w_{ij}\mathbf{y}_i\|^2$ which is equivalent to minimize $\mathrm{tr}(YMY^\top)$, where $M = (I-W)^\top(I-W)$. The M matrix can also be partitioned in a similar way as:

$$M = \begin{pmatrix} M_{ll} & M_{lu} \\ M_{lu}^\top & M_{uu} \end{pmatrix}$$

Replacing Y and M with a partitioned version, the objective function in LLE can be rewritten as :

$$\min_{Y_u} \quad \mathrm{tr}\left(\begin{pmatrix} Y_l & Y_u \end{pmatrix} \begin{pmatrix} M_{ll} & M_{lu} \\ M_{lu}^\top & M_{uu} \end{pmatrix} \begin{pmatrix} Y_l^\top \\ Y_u^\top \end{pmatrix} \right) \tag{2.33}$$

which is equivalent to

$$\min_{Y_u} \quad \mathrm{tr}\left(Y_u M_{uu} Y_u^\top + 2 Y_l M_{lu} Y_u^\top \right) \tag{2.34}$$

The solution to the above objective function can be computed by solving a linear system of equations as:

$$M_{uu} Y_u^\top = M_{lu} Y_l^\top \tag{2.35}$$

Semi-supervised ISOMAP For the semi-supervise version of ISOMAP [170], we first restate the ISOMAP problem as:

$$\begin{aligned} \max_{Y} \quad & \mathrm{tr}\left(YAY^\top \right) \\ \text{s.t.} \quad & YY^\top = I \end{aligned} \tag{2.36}$$

where A is the matrix as:

$$A = -\frac{1}{2}(I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top)^\top \Delta (I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top)$$

in which, Δ is the matrix of squared geodesic distances.

Let $A = Q\Lambda Q^\top$ is the eigen decomposition of A. $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ is a diagonal matrix where λ_i is the eigenvalue with $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$. Matrix Q consists of $Q = [\mathbf{q}_1, \ldots, \mathbf{q}_n]$ and $\mathbf{q}_i$ is the eigenvector corresponding to λ_i. Define matrix M as:

$$M = \lambda_1 I - A - \sum_{i=2}^{r} (\lambda_1 - \lambda_i)\mathbf{q}_i\mathbf{q}_i^\top - \lambda_1 \mathbf{1}\mathbf{1}^\top / n \tag{2.37}$$

It can be shown that matrix M has $r+1$ zero eigenvalues and the null space is given by $span([\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_r, \mathbf{1}])$. Therefore, (2.36) can be rewritten as:

$$\begin{aligned} &\min_{Y} \quad \text{tr}\left(YMY^\top\right) \\ &\text{s. t.} \quad YY^\top = I,\ \sum_{i=0}^{n} \mathbf{y}_i = \mathbf{0} \end{aligned} \tag{2.38}$$

Using the same idea of partitioning as semi-supervised LLE, we can incorporate the supervised information into the optimization problem in (2.38). As a result, this yields the same format as (2.33), except the matrix M is replaced by (2.37).

2.5 Functional Data Dimensionality Reduction

The functional data dimensionality reduction [74] is inspired by the Functional Data Analysis (FDA) [123]. The FDA approach is based on the assumption that multivariate data are discrete samples of some underlying functions and further analysis is carried out with the functional representations instead of the original data. The advantage is that, provided that the data is smooth enough, the functional representation will be of smaller dimension. One typical approach to obtain the functional representation is to construct a finite dimensional function space and then represent the data as the linear combination of a set of basis functions which span that space. The dimension of the functional representation of the data thus becomes the dimension of the combination weighting vectors which could be lower than the original data dimension if only a limited number of basis functions are used. However,

the choice of basis is not trivial because in many real-world applications the basis has a severe impact on the overall performance, therefore it is often desirable to construct problem specific basis functions to get good representation with a small number of basis functions while maintaining most of the original information. The shortcoming of the functional data dimensionality reduction is that it is more restricted because one has to assume that the multidimensional data can be represented in functional form.

The central idea of functional data analysis[123] is to treat high dimensional data as continuous functions. In general, the functions lie in a space with infinite dimensions and in order to make the computation tractable, the functions are usually approximated with finite dimensional representations. More specifically, it assumes that for each datum $\mathbf{x}_i = [x_i^1, x_i^2, \ldots, x_i^m]^\top$, there exists a function $f_i(z)$ belonging to certain function space $\mathcal{L}$ such that $x_i^h = f_i(z_h) + \epsilon_i^h$ for all $h = 1, \ldots, m$, where ϵ_i^h is the noise and the argument z_h where the observation are made are problem specific. The idea is to estimate the underlying function f_i based on the available data and conduct further analysis with the function representation instead of the original discrete data.

In general, the function space where f_i is defined is of infinite dimensions making it very difficult to work with the function f_i directly. In practice, it is very often to approximate f_i with a finite dimensional representation w_i which is defined in some r dimensional subspace $\mathcal{A} \subset \mathcal{L}$. Since the goal of this whole procedure is to reduce the dimensionality, a natural requirement is that the dimension of the subspace r is less than the original data dimension which is m.

The subspace is defined by a set of basis functions $\psi_j(z), j = 1, \ldots, r$ which span $\mathcal{A}$. Given the basis, any function f_i in $\mathcal{A}$ can be represented by the weight vector $\mathbf{w}_i = [w_i^1, w_i^2, \ldots, w_i^r]^\top$:

$$f_i(z) = \mathbf{w}_i^\top \boldsymbol{\psi}(z), \tag{2.39}$$

where $\boldsymbol{\psi}(z) = [\psi_1(z), \psi_2(z), \ldots, \psi_r(z)]^\top$. One typical approach to obtain the weights is to

minimize the square fitting error:

$$\min_{\mathbf{w}_i} \quad \sum_{l=1}^{m} \left(\mathbf{w}_i^\top \boldsymbol{\psi}(z_l) - x_i^l \right)^2 . \tag{2.40}$$

In order to obtain more stable models, regularization is often applied to (2.40), for example, a penalty term $\lambda \|\mathbf{w}_i\|^2$ can be added to (2.40) where λ is introduced to balance between the regularization and the regression error.

In general, orthogonal basis, such as Fourier basis or wavelets are often used, however, there are no free parameters and thus the basis cannot be optimized for a specific task. It is sometimes appealing to use non-orthogonal functions as basis so that a small number of weights are needed for representing the data. For instance, we can use Gaussian basis functions:

$$\psi_j(z) = \exp\left(\frac{-\|z - r_j\|^2}{\sigma_j^2} \right), j = 1, \ldots, m \tag{2.41}$$

where r_j is the location and σ_j is the width of the function. Clearly, these functions are not orthonormal. The basis is differentiable respect to the parameters r_j and σ_j, thus it can be optimized for an accurate fitting using standard gradient-based optimization algorithms. For example, if we choose the Gaussian basis functions and minimize the square fitting error of all the functions from $i = 1, \ldots, n$ as:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} (\Psi \mathbf{w}_i - \mathbf{x}_i)^\top (\Psi \mathbf{w}_i - \mathbf{x}_i) \\ = \quad & \sum_{i=1}^{n} \left(\mathbf{w}_i^\top \Psi^\top \Psi \mathbf{w}_i - 2\mathbf{x}_i^\top \Psi \mathbf{w}_i + \mathbf{x}_i^\top x_i \right), \end{aligned} \tag{2.42}$$

where the columns of Ψ are $\Psi_j = [\psi_j(z_1), \psi_j(z_2), \ldots, \psi_j(z_m)]^\top$ and it's derivative with respect to r_j and σ_j are:

$$\Psi_j^{(r)} = \left[\frac{z_1 - r_j}{\sigma_j^2} \psi_j(z_1), \ldots, \frac{z_m - r_j}{\sigma_j^2} \psi_j(z_m) \right]^\top \tag{2.43}$$

$$\Psi_j^{(\sigma)} = \left[\frac{(z_1 - r_j)^2}{\sigma_j^3} \psi_j(z_1), \ldots, \frac{(z_m - r_j)^2}{\sigma_j^3} \psi_j(z_m) \right]^\top \tag{2.44}$$

With this notation, it is easy to derive:

$$\frac{\partial \left(\mathbf{x}_i^\top \Psi \mathbf{w}_i \right)}{\partial r_j} = \mathbf{x}_i^\top \Psi_j^{(r)} \mathbf{w}_i^j \tag{2.45}$$

$$\frac{\partial \left(\mathbf{w}_i^\top \Psi^\top \Psi \mathbf{w}_i \right)}{\partial r_j} = 2\mathbf{w}_i^\top \Psi^\top \Psi_j^{(r)} \mathbf{w}_i^j \tag{2.46}$$

and finally we can get:

$$\frac{\partial}{\partial r_j} = \sum_{i=1}^{n} \left(\Psi \mathbf{w}_i - \mathbf{x}_i \right)^\top \Psi_j^{(r)} \mathbf{w}_i^j \tag{2.47}$$

Similarly, we can derive:

$$\frac{\partial}{\partial \sigma_j} = \sum_{i=1}^{n} \left(\Psi \mathbf{w}_i - \mathbf{x}_i \right)^\top \Psi_j^{(\sigma)} \mathbf{w}_i^j \tag{2.48}$$

After the gradient is computed, we can obtain the optimal r_i and σ_j using standard unconstrained non-linear optimization.

2.6 Summary

Dimensionality reduction is a board area and our review is far from complete. Here, we only focus on the methods of dimensionality reduction that are most widely used. There are many other well known algorithms for dimensionality reduction that are not included in the disscussion, such as Projection Pursuit [73], Principal Curve [57], Topologically Continuous Surfaces [21], Neural Network [68], and Independent Component Analysis [65]. For reviews of these methods please refer to [45] [21].

Recent studies have revealed interesting connection between dimensionality reduction and data clustering. For example, studies have shown that PCA is closely related to K-means[34], and non-negative matrix factorization is closely related to spectral clustering [35]. Also, studies [4] have shown that Laplacian eigenmaps [4] and spectral clustering [142], although not mathematically equivalent, share very similar ideas and only differ in the last one step: the last step of clustering thresholds the elements y_i in order to to divide the data points into two disjointed sets, while in dimensionality reduction the obtained matrix Y becomes the reduced representation of the original data patterns. In [36] the authors further formulate the problem of multi-way spectral clustering into the dimensionality reduction problem of finding the best one dimensional embedding, thus bridging gap between clustering and dimensionality reduction.

Although many methods have been proposed and successfully applied to real-world problems in reducing dimensionality, several open problems need further study. For example, most non-linear methods are not applicable for very large scale data sets ($n > 100,000$). Another issue is that the "target" dimension r often needs to be predetermined. However, determining the right target dimension is, sometimes, an even more difficult problem than dimensionality reduction itself.

CHAPTER 3

Semi-Supervised Learning by Mixed Label Propagation

In this chapter, we aim to explore the dimensionality reduction method for semi-supervised classification via the idea of mixed label propagation. In particular, our work is inspired by [36] where a clustering problem is approximated by a low dimension embedding problem. We extend this idea by approximating a data classification problem into a dimensionality reduction problem: we attempt to find the best one dimensional embedding of the data in which data points in different classes can be well separated, and the class labels obtained by simply thresholding the one dimensional representation. We furthermore extend LDA, a supervised dimensionality reduction method, to the semi-supervised learning setting. In this chapter, we first briefly review the existing work on graph-based semi-supervised learning, which is closely related to our work. We then present the proposed framework of mixed label propagation that essentially extend dimensionality reduction methods to semi-supervised classification.

3.1 Graph-Based Semi-Supervised Learning

Recent studies have shown a promising performance of graph-based approaches for semi-supervised learning [1, 7, 25, 59, 70, 174, 175, 176]. The key idea behind most graph-based approaches is to explore the pair-wise similarity between examples in determining the class labels for unlabeled examples. In particular, the class assignments of unlabeled examples need to be consistent with both the example similarity and the class labels of

training examples. Graph-based approaches can often be interpreted as propagating the label information of the training examples to the unlabeled examples through the pair-wise similarity between examples. This process is sometimes referred to as label propagation [175].

One key component to most graph-based approaches is how to measure the inconsistency between the class assignments and the example similarity. For instance, in the harmonic function approach the inconsistency between the class assignment $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ and similarity $S_{ij} \geq 0$ is measured by the energy function:

$$E(S, \mathbf{y}) = \sum_{i,j=1}^{n} S_{i,j} \left(y_i - y_j\right)^2 = \mathbf{y}^\top L \mathbf{y} \tag{3.1}$$

where L is the Laplacian matrix and is defined as $L = D - S$. Here, $D = \mathrm{diag}(D_1, D_2, \ldots, D_n)$ is a diagonal matrix with its diagonal elements defined as $D_i = \sum_{j=1}^{n} S_{i,j}$. Note that (3.1) is almost the same as the objective function of Laplacian Eigenmap in Section 2.2.3. The only difference is, here the variable $\mathbf{y}$ is class labels, while in 2.2.3 it is the low dimensional representation of data.

Given the class labels $\hat{\mathbf{y}}_l = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{n_l})$ for the first n_l examples, the optimal class assignment $\mathbf{y}$ is found by minimizing the above energy function, i.e.,

$$\begin{aligned} \min_{\mathbf{y}} \quad & E(S, \mathbf{y}) \\ \text{s. t.} \quad & \mathbf{y}_l = \hat{\mathbf{y}}_l \end{aligned} \tag{3.2}$$

where $\mathbf{y}_l$ stands for the first n_l elements of $\mathbf{y}$. The optimal class labels assigned to the unlabeled examples, denoted by $\mathbf{y}_u$, are computed as:

$$\mathbf{y}_u = -[L^{u,u}]^{-1} L^{u,l} \hat{\mathbf{y}}_l \tag{3.3}$$

where the super indices u and l stand for the parts of the Lapacian matrix that are related to the labeled and the unlabeled examples, respectively. Similar strategy is also employed for semi-supervised dimensionality reduction reviewed in section 2.4.2

It is important to note that the pair-wise similarity S_{ij} in the above energy function must be non-negative. This is because $E(S, \mathbf{y})$ could become negatively unbounded when certain

pair-wise similarity $S_{i,j}$ is negative, which implies the optimal solution to (3.3) does not exist.

3.2 Mixed Label Propagation

Despite the success, most graph-based approaches are limited to exploring positive similarity, which can be interpreted as the confidence of assigning two different examples to the same class. In many cases we may run into dissimilarity or negative similarity that expresses the confidence of assigning two examples to different classes. For instance, if we measure the similarity between two examples by their correlation coefficient, we could have both positive and negative similarity. One application of negative similarity is collaborative filtering [17], in which a negative similarity between two users indicates that the two users share different interests and therefore tend to give opposite ratings for the same items. Another application of negative similarity is semi-supervised data clustering, in which one has side information of must-link pairs and must-not-link pairs. One way to explore the side information is to associate every must-link pair with a positive similarity, and every must-not-link pair with a negative similarity [82].

It is important to note that most existing graph-based approaches are unapplicable to negative similarity because the dissimilar relations are non-transitive and therefore cannot be propagated directly. This can also be understood from the viewpoint of optimization. The energy function, i.e., the objective function employed by most graph-based approaches to measure the inconsistency between the class assignments and the example similarity, could be negatively unbounded when similarity is negative, thus no optimal solution can be found to minimize the objective function. To address this problem we propose a new framework of label propagation for semi-supervised learning, termed as **mixed label propagation**, which can effectively explore both negative and positive similarity simultaneously. Mixed label propagation measures two quantities: the inconsistency between the class assignments and

the positive similarity, and the consistency between the class assignments and the negative similarity. The optimal class assignments are found by minimizing the ratio between the inconsistency and the consistency.

3.2.1 The Framework of Mixed Label Propagation

To incorporate negative similarity into the framework of label propagation, we consider constructing two energy functions: the energy function E_+ that measures the *inconsistency* between the class assignments and the positive similarity, and the energy function E_- that measures the *consistency* between the class assignments and the negative similarity. In order to minimize the inconsistency E_+ and maximize the consistency E_- simultaneously, we follow the idea of Linear Discriminative Analysis (LDA) [44] by minimizing the ratio between E_+ and E_-. More specifically, given the pair-wise similarity S, we construct the positive similarity matrix S_+ and the negative similarity matrix S_- as follows:

$$[S_+]_{i,j} = \begin{cases} S_{i,j} & S_{i,j} > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$[S_-]_{i,j} = \begin{cases} |S_{i,j}| & S_{i,j} < 0 \\ 0 & \text{otherwise} \end{cases}$$

Evidently, we have $S = S_+ - S_-$. We then construct two energy functions $E_+(S_+, \mathbf{y})$ and $E_-(S_-, \mathbf{y})$ based on the two similarity matrices S_+ and S_- using Eqn. (3.1). Finally, given the class labels $\hat{\mathbf{y}}_l \in \{-1, +1\}^{n_l}$ for the first n_l training examples, the optimal class assignment $\mathbf{y}$ is determined by minimizing the ratio between E_+ and E_-, i.e.,

$$\begin{array}{ll} \min\limits_{\mathbf{y} \in \mathbb{R}^n} & \dfrac{E_+(S_+, \mathbf{y})}{E_-(S_-, \mathbf{y})} = \dfrac{\mathbf{y}^\top L_+ \mathbf{y}}{\mathbf{y}^\top L_- \mathbf{y}} \\ \text{s. t.} & y_i = \hat{y}_i,\ i = 1, 2, \ldots, n_l \end{array} \tag{3.4}$$

where L_+ and L_- are graph Laplacians for similarity matrices S_+ and S_-, respectively. Note that without the linear constraints $y_i = \hat{y}_i, i = 1, 2, \ldots, n$, the above optimization problem is identical to the optimization problem in LDA [44]. Hence, the optimal solution $\mathbf{y}$ to (3.4) is the minimum eigenvector of matrix $L_-^\dagger L_+$ where $\dagger$ stands for the pseudo inverse. The

challenges of solving the optimization problem in (3.4) arises from the linear constraints. In the next subsection we present an efficient algorithm to solve this optimization problem.

3.2.2 An Efficient Algorithm for Mixed Label Propagation

For the convenience of presentation we rewrite class assignments $\mathbf{y}$ as $\mathbf{y} = (\mathbf{y}_l, \mathbf{y}_u)$, where $\mathbf{y}_l$ represents the class labels of the first n_l examples and $\mathbf{y}_u$ represents the labels for the next $n_u = n - n_l$ examples. According to the constraints in (3.4), we have $\mathbf{y}_l = \hat{\mathbf{y}}_l$.

To solve the problem in (3.4), we first follow the idea of LDA by converting the problem of optimizing a ratio into a constrained optimization problem, i.e.,

$$\begin{array}{ll} \min\limits_{\beta\in\mathbb{R},\mathbf{y}_u\in\mathbb{R}^{n_u}} & \mathbf{y}^\top L_+\mathbf{y} \\ \text{s. t.} & \mathbf{y}^\top L_-\mathbf{y} \geq 1, \quad \beta \geq 0 \end{array} \tag{3.5}$$

where

$$\begin{aligned} \mathbf{y}^\top L_+\mathbf{y} &= (\beta\hat{\mathbf{y}}_l^\top, \mathbf{y}_u^\top) \begin{pmatrix} L_+^{l,l} & L_+^{l,u} \\ L_+^{u,l} & L_+^{u,u} \end{pmatrix} \begin{pmatrix} \beta\hat{\mathbf{y}}_l \\ \mathbf{y}_u \end{pmatrix} \\ \mathbf{y}^\top L_-\mathbf{y} &= (\beta\hat{\mathbf{y}}_l^\top, \mathbf{y}_u^\top) \begin{pmatrix} L_-^{l,l} & L_-^{l,u} \\ L_-^{u,l} & L_-^{u,u} \end{pmatrix} \begin{pmatrix} \beta\hat{\mathbf{y}}_l \\ \mathbf{y}_u \end{pmatrix} \end{aligned}$$

Note that in (3.5) we introduce a scaling factor β for $\hat{\mathbf{y}}_l$. This is because we introduce the constraint $\mathbf{y}^\top L_-\mathbf{y} \geq 1$, and therefore have to convert $\mathbf{y}_l = \hat{\mathbf{y}}_l$ to $\mathbf{y}_l \propto \hat{\mathbf{y}}_l$. β is introduced to account for the scaling factor between $\mathbf{y}_l$ and $\hat{\mathbf{y}}_l$.

We take the alternating optimization strategy to solve (3.5). More specifically, we first optimize $\mathbf{y}_u$ by fixing β and then optimize β by fixing $\mathbf{y}_u$. However, the problem in (3.5) is a non-convex programming problem for both β and $\mathbf{y}_u$ because of the non-convex constraint $\mathbf{y}^\top L_-\mathbf{y} \geq 1$. To resolve this problem we resort to the following theorem of the alternative [15]:

Theorem 3.1. *The implication*

$$\mathbf{x}^\top F_1\mathbf{x} + 2g_1^\top\mathbf{x} + h_1 \leq 0 \Longrightarrow \mathbf{x}^\top F_2\mathbf{x} + 2g_2^\top\mathbf{x} + h2 \leq 0,$$

where F_i is symmetric $n \times n$ matrix, holds if and only if there exists $\lambda \geq 0$ such that

$$\begin{pmatrix} F_2 & g_2 \\ g_2^\top & h_2 \end{pmatrix} \succeq \lambda \begin{pmatrix} F_1 & g_1 \\ g_1^\top & h_1 \end{pmatrix}$$

Optimize y with fixed β Using the above theorem, to compute the optimal $\mathbf{y}_u$ with fixed β, we turn the problem in (3.5) into its dual form, i.e.,

$$\begin{aligned} \max_{\lambda} \quad & \lambda a_{22} - \beta^2 a_{21} a_{11}^{-1} a_{12} \\ \text{s. t.} \quad & \lambda \geq 0 \\ & a_{11} = L_+^{u,u} - \lambda L_-^{u,u}, a_{12} = \left(L_+^{u,l} - \lambda L_-^{u,l}\right) \hat{\mathbf{y}}_l \\ & a_{21} = \hat{\mathbf{y}}_l^\top \left(L_+^{l,u} - \lambda L_-^{l,u}\right), a_{22} = 1 - \beta^2 \hat{\mathbf{y}}_l^\top L_-^{l,l} \hat{\mathbf{y}}_l \end{aligned} \tag{3.6}$$

Given the solution for λ, we can compute the solution for $\mathbf{y}_u$ using the Karush-Kuhn-Tucker (KKT) conditions [15], i.e.,

$$\mathbf{y}_u = -\beta \left(L_+^{u,u} - \lambda L_-^{u,u}\right)^{-1} \left(L_+^{u,l} - \lambda L_-^{u,l}\right) \mathbf{y}_l \tag{3.7}$$

It is interesting to note that the above solution for $\mathbf{y}_u$ is equivalent to the solution by the harmonic function (in Eqn. (3.3)) if we use $L = L_+ - \lambda L_-$ as the graph Laplacian matrix. Thus, the parameter λ weighs the importance between the two energy functions E_+ and E_-. The advantage of the proposed approach is that it automatically determines λ by making the optimal tradeoff between the inconsistency measure E_+ and the consistency measure E_-. This is particularly important for semi-supervised learning when the number of labeled examples is limited and is insufficient to determine λ by cross validation. We will also show in our empirical study that the value of λ varies significantly from one case to another, and therefore it is suboptimal to replace λ with a fixed constant. The problem in (3.6) can be further turned into a Semi-Definite Programming (SDP) [156](See Appendix C) problem as

follows:

$$\begin{aligned} \max_{\lambda,\gamma} \quad & \gamma \\ \text{s. t.} \quad & \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & (\lambda a_{22} - \gamma)/\beta^2 \end{pmatrix} \succeq 0,\ \lambda \geq 0 \\ & a_{11} = L_+^{u,u} - \lambda L_-^{u,u},\ a_{12} = \left(L_+^{u,l} - \lambda L_-^{u,l}\right)\hat{\mathbf{y}}_l \\ & a_{21} = \hat{\mathbf{y}}_l^\top \left(L_+^{l,u} - \lambda L_-^{l,u}\right), a_{22} = 1 - \beta^2 \hat{\mathbf{y}}_l^\top L_-^{l,l} \hat{\mathbf{y}}_l \end{aligned} \tag{3.8}$$

In (3.8) we introduce the slack variable $\gamma \leq \lambda a_{22} - \beta^2 a_{21} a_{11}^{-1} a_{12}$, which can be further turned into a Linear Matrix Inequality (LMI)

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & (\lambda a_{22} - \gamma)/\beta^2 \end{pmatrix} \succeq 0$$

using the Schur complement [15](see Appendix B). Since the problem in (3.8) belongs to the family of semi-definitive programming, it can be solved effectively using the standard packages such as SeDuMi[1].

Optimize β with fixed y Similarly, using the theorem 1 we have the following optimization problem for finding optimal β with fixed $\mathbf{y}_u$

$$\begin{aligned} \max_{\lambda,\gamma} \quad & \gamma \\ \text{s. t.} \quad & \lambda \geq 0,\ \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} \succeq 0 \\ & b_{11} = \hat{\mathbf{y}}_l^\top L_+^{l,l} \hat{\mathbf{y}}_l - \lambda \hat{\mathbf{y}}_l^\top L_-^{l,l} \hat{\mathbf{y}}_l \\ & b_{12} = b_{21} = \hat{\mathbf{y}}_l^\top L_+^{l,u} \mathbf{y}_u - \lambda \hat{\mathbf{y}}_l^\top L_-^{l,u} \mathbf{y}_u \\ & b_{22} = -\lambda \left(\mathbf{y}_u^\top L_-^{u,u} \mathbf{y}_u - 1\right) - \gamma \end{aligned} \tag{3.9}$$

The corresponding solution for β is

$$\beta \quad = \quad -\frac{\hat{\mathbf{y}}_l^\top L_+^{l,u} \mathbf{y}_u - \lambda \hat{\mathbf{y}}_l^\top L_-^{l,u} \mathbf{y}_u}{\hat{\mathbf{y}}_l^\top L_+^{l,l} \hat{\mathbf{y}}_l - \lambda \hat{\mathbf{y}}_l^\top L_-^{l,l} \hat{\mathbf{y}}_l} \tag{3.10}$$

[1] `http://sedumi.mcmaster.ca/`

In summary, we start with a large value for β, then find the optimal $\mathbf{y}_u$ by solving the problem in (3.8) with a fixed β, and find the optimal β by solving the problem in (3.9) with a fixed $\mathbf{y}_u$. We alternate these two steps iteratively until the solution converges to the local maximum.

3.2.3 Application to Collaborative Filtering

In order to verify the efficacy of the proposed algorithms for semi-supervised learning, we apply it to collaborative filtering. We emphasize that we choose collaborative filtering for the test bed because we are able to compute both similarity and dissimilarity in the task of collaborative filtering, a key requirement for the proposed method. As discussed in Section 2.3.4, the goal of collaborative filtering is to predict the utility of items to a user based on ratings by other users [126]. This method differs from the traditional feature based approach where predictions are made based on the features of data. For example, in a collaborative filtering movie recommendation system[2], features of the movies (e.g., genre, year, actors, external reviews) and the users (e.g., age, gender, explicitly specified preferences) are not used by the system. Inputs to the system are user ratings on movies they have already seen. When making predictions of user ratings on movies they have not yet seen, the system assume that users "collaborate" by sharing their ratings, e.g., to predict the ratings of a movie by user u, most collaborative filtering algorithms first identify a subset of users who share similar interests to u, and then combine the ratings of these similar users as the ratings by u. The most well known algorithms for collaborative filtering include the Pearson correlation coefficient [126], personality diagnosis [117], matrix factorization [147], graphical models [17, 61], and ordinal regression [25].

In order to apply the proposed approach to collaborative filtering, the key is to estimate the matrix S for user similarity. To this end, we employ the Pearson correlation coefficient [126] to estimate user similarity. It measures the linear correlation between the ratings of two

[2] `http://movielens.org/`

	Given 10 Rated Items	
	$K_c = 1$	$K_c = 5$
MLP	**0.8318** ± 2.7E-04	**0.7368** ± 1.2E-04
LP	0.8184 ± 8.6E-04	0.7316 ± 1.3E-04
Pearson	0.7766 ± 6.8E-04	0.6749 ± 4.0E-04
MMMF	0.7827 ± 9.2E-05	0.6974 ± 3.6E-05
	Given 15 Rated Items	
	$K_c = 1$	$K_c = 5$
MLP	**0.8599** ± 3.4E-04	**0.7704** ± 1.2E-04
LP	0.8526 ± 5.3E-04	0.7689 ± 1.4E-04
Pearson	0.8170 ± 1.0E-03	0.7222 ± 3.8E-04
MMMF	0.8189 ± 2.0E-04	0.7221 ± 8.4E-05

Table 3.1. Average precision for the Mixed Label Propagation (MLP), Label Propagation (LP), Pearson Correlation Coefficient (Pearson) and Maximum-Margin Matrix Factorization(MMMF) using 10 training users.

users. More specifically, given two users u_i and u_j, let $\mathcal{O}^{i,j}$ denote the set of items that are rated by both users. The similarity between u_i and u_j is measured as:

$$S_{i,j} = \frac{\sum_{k=1}^{|\mathcal{O}^{i,j}|}(r_i(k) - \bar{r}_i)(r_j(k) - \bar{r}_j)}{\sqrt{\sum_{k=1}^{|\mathcal{O}^{i,j}|}(r_i(k) - \bar{r}_i)^2}\sqrt{\sum_{k=1}^{|\mathcal{O}^{i,j}|}(r_j(k) - \bar{r}_j)^2}}$$

where $r_i(k)$ stands for the rating of the kth item by user u_i, and $\bar{r}_i$ is the average rating of user u_i. Since the Pearson correlation coefficient can be both negative and positive, we can apply the proposed method to collaborative filtering.

3.3 Experiments

We evaluated the effectiveness of the proposed mixed label propagation approach by the task of collaborative filtering and used a subset of the MovieLens database (`http://movielens.umn.edu/login`) as the test bed. In particular, we selected the 100 most popular movies and randomly selected 200 users who had provided at least 25 ratings for the 100 selected movies.

	Given 10 Rated Items	
	$K_c = 1$	$K_c = 5$
MLP	**0.8325** ± 3.7E-04	**0.7449** ± 1.3E-04
LP	0.8228 ± 1.3E-04	0.7408 ± 4.7E-05
Pearson	0.7998 ± 3.5E-04	0.6938 ± 3.4E-04
MMMF	0.7946 ± 1.5E-04	0.7036 ± 1.2E-04
	Given 15 Rated Items	
	$K_c = 1$	$K_c = 5$
MLP	**0.8661** ± 8.8E-05	**0.7732** ± 5.2E-05
LP	0.8592 ± 4.7E-04	0.7727 ± 1.4E-04
Pearson	0.8177 ± 3.3E-04	0.7291 ± 1.9E-04
MMMF	0.8128 ± 2.3E-05	0.7219 ± 1.5E-04

Table 3.2. Average precision for the Mixed Label Propagation (MLP), Label Propagation (LP), Pearson Correlation Coefficient (Pearson) and Maximum-Margin Matrix Factorization(MMMF) using 20 training users.

In contrast to the binary label requirement in (3.4), the labels here were the ratings ranging from 1 to 5, and the mixed label propagation algorithm was applied to each movie separately to predict the ratings by all users. To acquire a full spectrum of the performance, we varied the number of training users and the number of movies whose ratings were provided by the test users. More specifically, 10 and 20 users were used as the training users. For each test user, 10 and 15 movies were randomly selected and their ratings by the test user were given. Each experiment was conducted ten times, and the results averaged over ten trials were reported in our study.

Three baseline models were used in our study: the Pearson correlation coefficient method (Pearson) [126], the Maximum-Margin Matrix Factorization method (MMMF) [124], and the Label Propagation method (LP) [176] that is based on the harmonic function and only uses the positive similarity. By comparing to the Pearson correlation method and the maximum-margin matrix factorization method, we are able to observe if the proposed method is effective for collaborative filtering. By comparing to the label propagation method, we are able to

observe if the proposed method is effective in exploiting the negative similarity. For the maximum margin matrix factorization method, we used coding from the website `http://people.csail.mit.edu/nati/mmmf/code.html` and used a maximum norm with $C = 0.001$ for all the experiments following the suggestion in [147].

To examine the quality of different collaborative filtering algorithms, we focused on evaluating how well each method was able to rank items for users. For each user we ranked items in the descending order of their estimated ratings. We then evaluated the ranked items by comparing to the items ranked by the true ratings. More specifically, for a test user u, we used $\mathbf{l}_e = (i_1, i_2, \ldots, i_m)$ to denote the list of items ordered by the estimated ratings, and r_i^u to denote the true rating of the ith item by user u. Then, the quality of the ranked items was evaluated by the following two metrics:

- *Average Precision* [46] (AP). To measure the average precision, we assume that the first k items with the highest ratings by user u, denoted by $\mathcal{M}_u(k)$, are the "good" items for user u. Then, the average precision for user u at the cutoff rank K_c is computed as:

$$AP_u(K_c) = \frac{1}{K_c} \sum_{k=1}^{K_c} \frac{|\mathcal{M}_u(k) \cap \{i_1, \cdots, i_k\}|}{k} \tag{3.11}$$

  where $|\cdot|$ outputs the length of the set.

- *Average Rating* (AR). This computes the average rating of the first K_r ranked items in the list $\mathbf{l}_e$. More specifically, the average rating for user u at the rank K_r is computed as follows

$$AR_u(K_r) \quad = \quad \frac{1}{K_r} \sum_{k=1}^{K_r} r_{i_k}^u \tag{3.12}$$

Finally, we averaged both metrics over all test users, and reported the average precision at a different cutoff rank K_c and the average rating at a different ranking position K_r. Note that we did not use the Mean Average Error (MAE) [17] because MAE requires an additional step to calibrate scores into rating, and therefore does not directly reflect the quality of collaborative filtering algorithms.

3.3.1 Experiment (I): Effectiveness of Mixed Label Propagation

The average precision of the four methods for 10 and 20 training users are reported in Table 3.1 and 3.2, respectively. Figure 3.1 and 3.2 show the average ratings of the four methods for 10 and 20 training users, respectively. First, we observe that for all of the four methods, both the average precision and the average rating are improved with the larger number of training users and given more rated items. This is consistent with our expectation: the more the training examples, the better the performance. Second, according to the average precision metric, we observe that compared to other methods the Pearson correlation coefficient method achieves almost the worst performance. It is also surprising to observe that the maximum margin matrix factorization (i.e., MMMF) does not improve the prediction accuracy in comparison to the Pearson correlation coefficient. We believe that this may be attributed to the small number of training users in our study while most of the previous studies of MMMF focused on large numbers of training users. Third, the label propagation method based on the harmonic function performs better than both the Pearson correlation coefficient method and the maximum-margin matrix factorization method according to the average precision metric. Finally, according to both metrics, the mixed label propagation method outperforms the other three methods considerably in all experiments and the improvements are statistical significant under the student-t test with 95% confidence interval.

3.3.2 Experiment (II): Empirical Values for λ

As described before, the solution for $\mathbf{y}_u$ in Eqn. (3.7) is essentially equivalent to the solution by the harmonic function (in Eqn. (3.3)) if we use $L = L_+ - \lambda L_-$ as the graph Laplacian matrix. Since we have to solve the mixed label propagation problem for each movie, we compute a different λ for each movie. Figure 3.3 shows the λs that were computed for the 100 movies with 20 training users and 10 given ratings. We clearly see the large variance in λ across different movies. For some movies the optimal λ can be as high as 0.3. For other movies the optimal λ can be as low as 10^{-4}. Given the empirical values of λ shown

in Figure 3.3, it is unlikely to find a fixed λ that can fit in well with all movies. This is also confirmed by our empirical study with fixed λ.

3.4 Discussion

A closely related work [48] was published almost the same time as our mixed label propagation, here we refer it to the mixed-graph method. In the mixed-graph method, the authors try to incorporate both similarity and dissimilarity into semi-supervised classification problems. More specifically, they define a mixed graph of n nodes by two $n \times n$ matrices S and W. The matrix S specifies the edge type: $S_{ij} = 1$ if there is a similarity between ith node and jth node; $S_{ij} = -1$ if there is a dissimilarity edge. Non-negative weights $w_{ij} \geq 0$ represent the weights of the edge between ith node and jth node regardless of its type.

In the next step, they define matrix M as an analog of the graph Laplacian L as:

$$M \quad = \quad L + (\mathbf{1}\mathbf{1}^\top - S) \bullet W$$

where $\bullet$ stands for the elementwise product. Like the Laplacian matrix, M is also positive semi-definite. If the graph has no dissimilarity edges, M degenerates to L.

The M matrix is then used in the framework of manifold regularization [7], in which the discriminant function is obtained by solving:

$$\min_{f \in \mathcal{H}} \quad \sum_{i=1}^{l} c\left(y_i, f(\mathbf{x}_i)\right) + \lambda_1 \|f\|_{\mathcal{H}}^2 + \lambda_2 \mathbf{f}^\top M \mathbf{f} \tag{3.13}$$

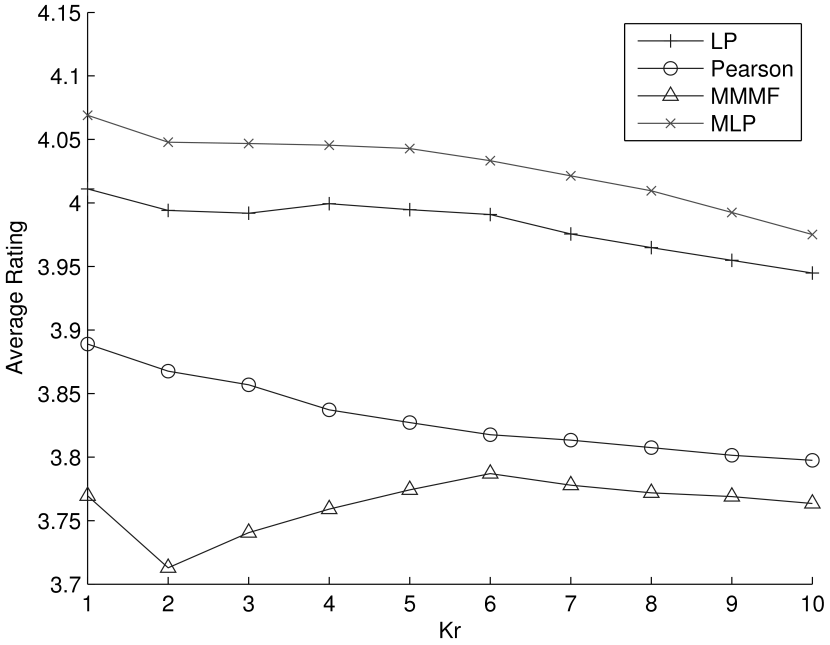
where $\mathcal{H}$ is the Reproducing Kernel Hilbert Space of a kernel K, $c()$ is an arbitrary loss function, e.g., the hinge loss for Support Vector Machine (SVM) [18], or squared loss for Regularized Least Squares classifiers (RLS) [41], and $\mathbf{f}$ is the vector of discriminant function values on the n points. The first two terms are the same as in supervised learning, while the third term is the additional regularization term for the mixed-graph semi-supervised learning.

Comparing to our mixed label propagation, instead of constructing energy functions associated with similarity and dissimilarity separately, the mixed-graph method tries to find a unified way to express both similarity and dissimilarity, thus resulting a mixed version of the standard Laplacian matrix which is the M matrix. This, however, brings both advantages and disadvantages. First, the positive side is that the M matrix is still positive semi-definite and this makes the optimization problem of (3.13) to be convex. The negative side is that the importance between the similarity and dissimilarity cannot be balanced automatically. One has to resort to other techniques, such as cross validation, to decide how important the dissimilarity is before the construction of matrix M. This can be seen clearly from the experiment part of the mixed-graph method in which the weights of dissimilarity are manually varied to exam the importance of dissimilarity. Our method, however, has the benefit of automatically deciding the importance between similarity and dissimilarity. As we discussed when explaining (3.7), our method essentially uses $L = L_+ - \lambda L_-$, and can automatically determine λ by making the optimal tradeoff between the inconsistency measure E_+ and the consistency measure E_-.

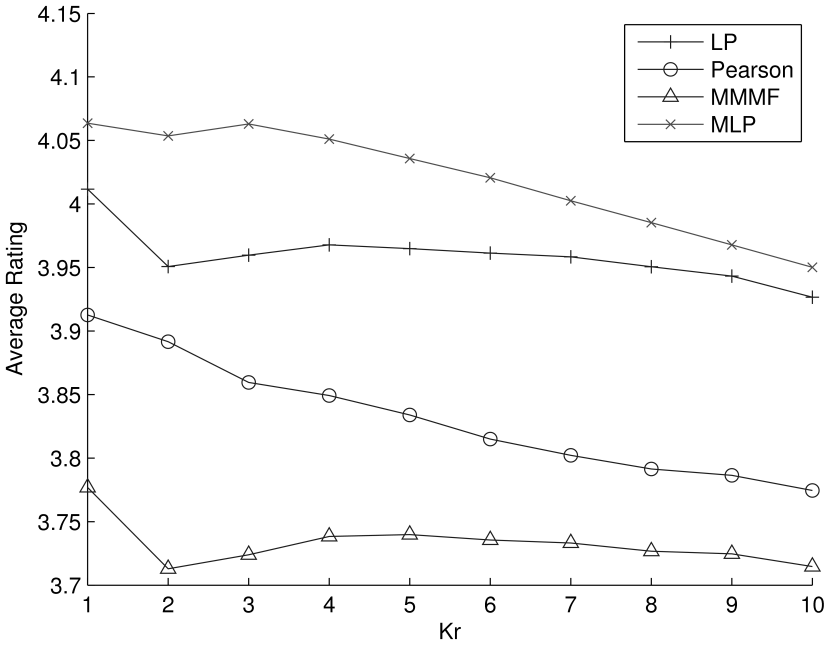
3.5 Summary

In this chapter, we proposed a mixed label propagation framework for semi-supervised learning. Unlike the existing graph-based approaches that are only applicable to the positive similarity of examples, our framework is able to explore both positive and negative similarity simultaneously. The key idea behind the proposed framework is very similar to LDA, which is to minimize the inconsistency between the class assignments and the positive similarity of examples, and maximize the consistency between the class assignments and the negative similarity of examples. We presented an efficient learning algorithm for the mixed label propagation that is based on the alternative optimization strategy and semi-definitive programming. Our empirical study with collaborative filtering showed that the proposed

algorithm is effective in exploring negative similarity and outperforms both the label propagation approach and state-of-the-art approaches for collaborative filtering.

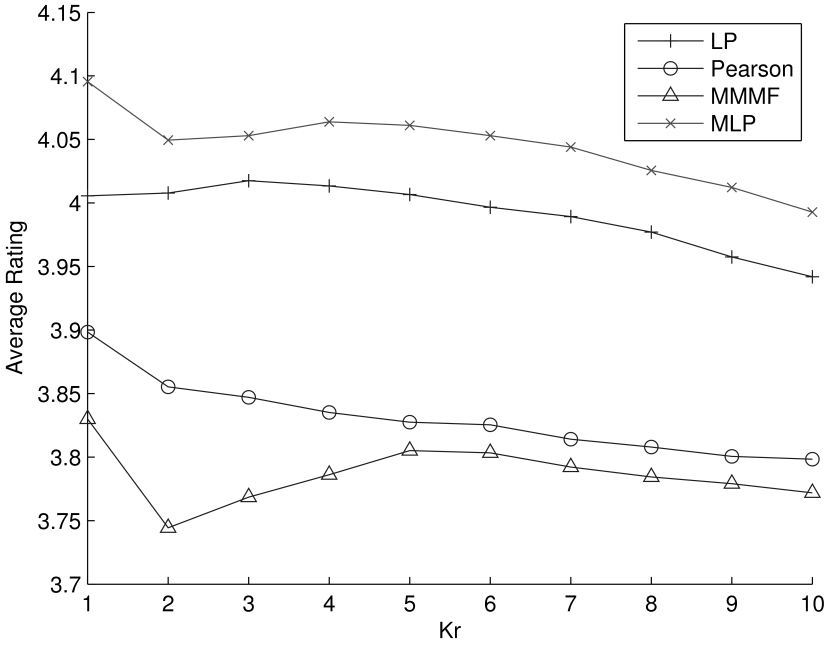


(a) 10 given rated items

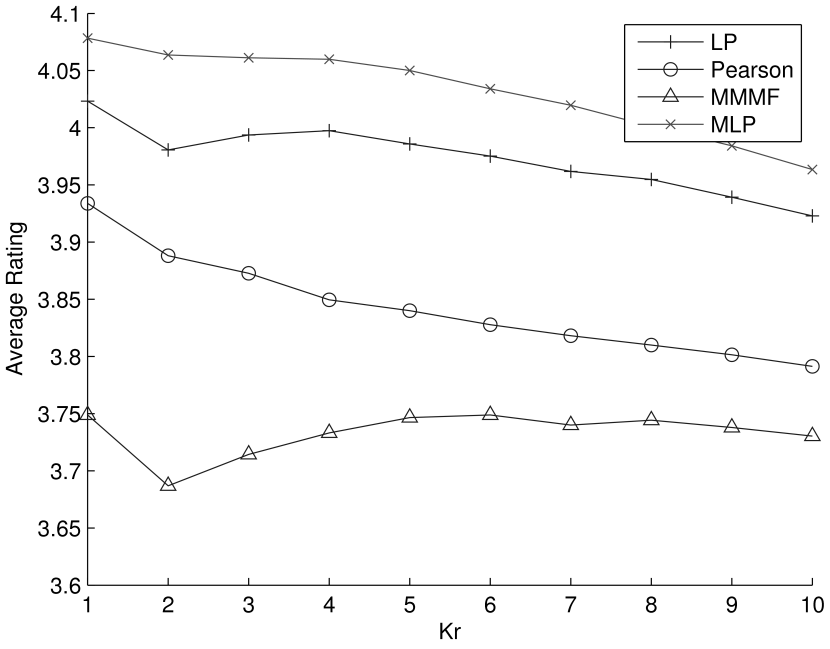


(b) 15 given rated items

Figure 3.1. The Average rating of the Mixed Label Propagation (MLP), Label Propagation (LP), Pearson Correlation Coefficient (Pearson) and Maximum-Margin Matrix Factorization(MMMF) using 10 training users.



(a) 10 given rated items



(b) 15 given rated items

Figure 3.2. The Average rating of the Mixed Label Propagation (MLP), Label Propagation (LP), Pearson Correlation Coefficient (Pearson) and Maximum-Margin Matrix Factorization(MMMF) using 20 training users.

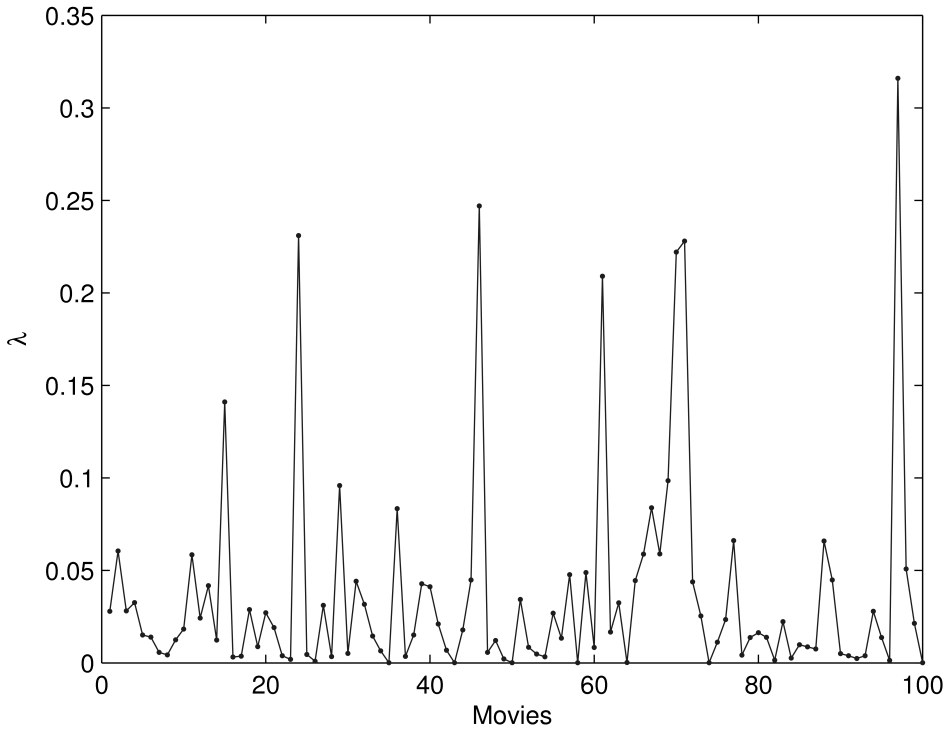


Figure 3.3. The λs computed for the 100 movies with 20 training users and 10 given ratings.

CHAPTER 4

Second-Order PCA-Style Dimensionality Reduction Algorithm by Semi-Definite Programming

PCA is one of the most widely used methods for dimensionality reduction. However, when handling very high dimensional data, such as images, PCA could be computationally expensive since it requires compute principle eigenvectors of a very large matrix. Recently, researchers proposed the Second-Order PCA-Style (SOPCA) algorithms [168, 172, 38, 20, 12, 95] that aim to alleviate the high computational cost of PCA. Unlike PCA that represents each object, such as an image, by a vector, these methods represent each datum by a matrix, a natural representation for images, which is the key for reducing the computational cost. Despite the success, almost all SOPCA methods require solving a non-convex optimization problem and as a result, are only able to identify the local optimal solution either by an alternative method or by an approximate approach. In this chapter we aim to alleviate this problem by presenting a convex optimization framework for SOPCA.

4.1 Second-Order PCA-Style Algorithms

PCA, or more general PCA-style methods, are based on the vector space model in which each datum is a vector and the collection of data is a matrix. There are two primary drawbacks when applying the vector space model to image data analysis. First, by converting each

image into a vector, the spatial relationship between neighboring pixels is completely lost leading to algorithms that are unable to fully utilize the spatial correlation of pixels. Second, due to the large number of pixels, each image has to be represented by a vector in a very high dimensional space. As a result, running PCA-style algorithms against images could be expensive both computationally and space wise due to the time and space complexities of singular value decomposition (SVD) for large matrices.

Early work aimed to reduce the computational complexity of PCA by incremental algorithms, such as [16, 53]. More recently, researchers have proposed the second-order PCA-style (SOPCA) algorithms [168, 172, 38, 20, 12, 95] to alleviate the computational complexity of PCA for images. These algorithms represent an image by a matrix, and directly conduct dimensionality reduction over a collection of matrices. By avoiding vector representation of images, these algorithms are able to achieve substantially better computational efficiency than PCA-style algorithms for image data analysis. Both theoretical and experimental results have shown that these methods have substantially lower computational cost than classical PCA-style algorithms when applied to high dimensional data.

One of the most representative works of SOPCA is the Generalized Low Rank Approximations of Matrices (GLRAM) [172]. It reduces the dimensionality of a matrix by multiplying it with left and right projection matrices. More specifically, we denote $\mathcal{M} = (M_1, M_2, \ldots, M_n)$ as the collection of matrices for dimensionality reduction, where each $M_i \in \mathbb{R}^{p\times q}$. The goal is to identify two matrices $L \in \mathbb{R}^{p\times k_1}$ and $R \in \mathbb{R}^{q\times k_2}$ such that each matrix M_i can be well approximated by $LA_iR^\top$, where $A_i \in \mathbb{R}^{k_1\times k_2}$. Often in practice, we set $k_1 = k_2 = k$, where k is the target dimension to be reduced. The L and R matrices are obtained by solving the following optimization problem:

$$\begin{aligned} \min_{L,R,A_i} \quad & \sum_{i=1}^{n} \|M_i - LA_iR^\top\|_F^2 \\ \text{s. t.} \quad & L^\top L = R^\top R = I_k. \end{aligned} \tag{4.1}$$

It is well known that (4.1) is a non-convex optimization problem, and usually an iterative

procedure is employed to obtain the local optimal solution. More specifically, an initial L is obtained first, for example, an identity matrix. Then, the columns of the projection matrix R are computed from the eigenvectors corresponding to the k largest eigenvalues of the matrix $\sum_{i=1}^{n} M_i^\top L L^\top M_i$. With the computed R, we can then update L by computing the eigenvectors of the matrix $\sum_{i=1}^{n} M_i R^\top R M_i^\top$ and form the matrix L using the eigenvectors corresponding to the k largest eigenvalues. The procedure is repeated until convergence.

Inspired by GLRAM, [38] extends the traditional Singular Value Decomposition (SVD) to the 2-D form. In 2-D SVD, the row-row and column-column covariance matrices are first computed as follows:

$$\begin{aligned} F &= \sum_{i=1}^{n} (M_i - \bar{M})(M_i - \bar{M})^\top \\ G &= \sum_{i=1}^{n} (M_i - \bar{M})^\top (M_i - \bar{M}), \end{aligned}$$

where $\bar{M} = \sum_{i=1}^{n} M_i / n$ is the mean of all matrices. The left and right projection matrices are then computed from the leading eigenvectors of F and G. It can be shown that the 2-D PCA [168] is a special case of 2-D SVD.

Similar to 2-D SVD, Tensor-PCA [20] views each image as a point in a tensor space and tries to find the optimal tensor subspace such that in that subspace, the variance of the projected data is maximized. The optimization problem of Tensor-PCA, like GLRAM, is non-convex. In [20] the authors present an approximate algorithm that indeed gives the same solution as 2-D SVD [66].

In [12] the authors extend Probabilistic PCA [13], a probabilistic interpretation of PCA reviewed in Section 2.1.3, to matrix and tensor dimensionality reduction. The authors present a family of probabilistic models, termed Probabilistic Higher-Order PCA, that explicitly specify the generative process for higher-order objects such as matrices. Studies in [77, 95] also consider the extension of second-order PCA to high-order tensor objects.

4.2 A Convex Formulation for SOPCA

The primary difficulty with identifying the left and right projection matrices L and R in GLRAM arises from their dependency, namely, the solution of L depends on the solution of R and vice versa. The key observation motivating this work is that if we can factorize each matrix M_i in advance into a product of left matrix U_i and right matrix V_i, then the left and right projection matrices L and R could be roughly computed by examining the eigenvectors of the sum of the left and right matrices. Based on this motivation, we introduce a Semi-definite Programming (SDP) [156] formulation for SOPCA, which is a convex optimization problem and is guaranteed to find the global optimal solution.

4.2.1 Low Norm Approximation of a Single Matrix

The key question now is how to determine the appropriate factorization for M_i. In order to motivate the right formulation, we first consider the case with a single matrix X. Assume $X = U\Sigma V^\top$ is the singular value decomposition of X. According to the definition of the trace norm of a matrix [136](See Appendix D), the optimal solution to the following optimization problem:

$$\begin{aligned} \min_{\tilde{U},\tilde{V}} \quad & \mathrm{tr}(\tilde{U}\tilde{U}^\top) + \mathrm{tr}(\tilde{V}\tilde{V}^\top) \\ \text{s. t.} \quad & X = \tilde{U}\tilde{V}^\top \end{aligned} \tag{4.2}$$

satisfies the condition:

$$\begin{aligned} \tilde{U}\tilde{U}^\top &= U\Sigma U^\top \\ \tilde{V}\tilde{V}^\top &= V\Sigma V^\top \end{aligned}$$

This provides us with a good staring point for the factorization of a matrix X. In the next step, we extend (4.2) from extracting all singular vectors to only the singular vectors corresponding to the k largest singular values, which is summarized in the following theorem:

Theorem 4.1. *The optimal value to the following optimization problem*

$$\begin{aligned} \min_{\substack{P\in\mathbf{S}_+^p \\ Q\in\mathbf{S}_+^q}} \quad & S_k(P) + S_k(Q) \\ s.\ t. \quad & \begin{pmatrix} P & X \\ X^\top & Q \end{pmatrix} \succeq 0 \end{aligned} \tag{4.3}$$

is $2\sum_{i=1}^k \sigma_i$*, where* $S_k(\cdot)$ *is the sum of the* k *largest singular value of a matrix, and* $\sigma_i, i = 1, \ldots, k$ *are the largest* k *singular values of* X*. The first* k *eigenvectors* P *and* Q *are the first* k *left and right singular vectors of matrix* X*.*

Proof. First, we could write $S_k(P)$ as an optimization problem [158], i.e.,

$$S_k(P) = \max_{Z_P\in\mathbf{S}_+^p} \{\mathrm{tr}(Z_P P) : 0 \preceq Z_P \preceq I_p, \mathrm{tr}(Z_P) = k\}$$

For the convenience of presentation, we denote

$$\Delta_p(k) = \{Z \in \mathbf{S}_+^p : 0 \preceq Z \preceq I_p, \mathrm{tr}(Z) = k\},$$

and therefore represent $S_k(P)$ as

$$S_k(P) = \max_{Z_P\in\Delta_p(k)} \mathrm{tr}(Z_P P).$$

Using the above notations, the problem in (4.3) can be transferred into a min-max optimization problem, which is:

$$\begin{aligned} \min_{\substack{P\in\mathbb{R}_+^p \\ Q\in\mathbb{R}_+^q}} \max_{\substack{Z_P\in\Delta_p(k) \\ Z_Q\in\Delta_q(k)}} \quad & \mathrm{tr}(Z_P P) + \mathrm{tr}(Z_Q Q) \\ \text{s. t.} \quad & \begin{pmatrix} P & X \\ X^\top & Q \end{pmatrix} \succeq 0 \end{aligned} \tag{4.4}$$

Using the Von Newmann's lemma [111], we could switch minimization and maximization.

Furthermore, by constructing the dual problem for P and Q, we have (4.4) rewritten as:

$$\begin{aligned} \max_{Z_P, Z_Q, Z_X} \quad & \mathrm{tr}(Z_X^\top X + Z_X X^\top) \\ \text{s. t.} \quad & \begin{pmatrix} Z_P & Z_X \\ Z_X^\top & Z_Q \end{pmatrix} \succeq 0 \\ & 0 \preceq Z_P \preceq I_p, \mathrm{tr}(Z_P) = k \\ & 0 \preceq Z_Q \preceq I_q, \mathrm{tr}(Z_Q) = k \end{aligned}$$

We write X in its singular value decomposition, i.e., $X = U\Sigma V^\top$, and the objective function becomes $2\mathrm{tr}\left((U^\top Z_X V + V^\top Z_X^\top U)\Sigma\right)$. We furthermore replace the LMI constraint as

$$\begin{pmatrix} U^\top & -V^\top \end{pmatrix} \begin{pmatrix} Z_P & Z_X \\ Z_X^\top & Z_Q \end{pmatrix} \begin{pmatrix} U \\ -V \end{pmatrix} \succeq 0$$

which leads to the following LMI constraint

$$U^\top Z_P U + V^\top Z_Q V - V^\top Z_X^\top U - U^\top Z_X V \succeq 0$$

Define

$$A = U^\top Z_P U,\ B = V^\top Z_Q V,\ C = V^\top Z_X^\top U + U^\top Z_X V$$

we finally simplify the original problem as:

$$\begin{aligned} \max_{A,B,C} \quad & \mathrm{tr}(C\Sigma) \\ \text{s. t.} \quad & C \preceq A + B \\ & 0 \preceq A, B \preceq I_m \\ & \mathrm{tr}(A) = \mathrm{tr}(B) = k \end{aligned}$$

Evidently, the optimal solution for C is $2I_k$, when A and B are $I_m(k)$, where $I_m(k)$ is a $m \times m$ diagonal matrix with first k diagonal elements being one, and zeros for the remaining diagonal elements. Note that since we enlarged the solution space by relaxing the LMI constraint, what we obtained here is the upper bound of the optimal value of problem (4.3). However, the solutions to C, A and B are exactly in the original space, therefore, the optimal

value of problem (4.3) can reach its upper bound which is twice the sum of the first k singular values of X. Since A and B are $I_m(k)$, we have Z_P and Z_Q as constructed by the top k column vectors of U and V, i.e., the first k left and right singular vectors of X. □

4.2.2 Low Norm Approximation for Multiple Matrices

The result in Theorem 5.2 indicates an alternative way of extracting the top singular vectors from a given matrix. In order to extract the optimal projection matrices from a collection of matrices, it is natural to extend (4.3) to multiple matrices. In particular, we propose the following optimization formulation for SOPCA.

$$\begin{aligned} \min_{\substack{P_i \in \mathbf{S}_+^p \\ Q_i \in \mathbf{S}_+^q}} \quad & S_k\left(\sum_{i=1}^n P_i\right) + S_k\left(\sum_{i=1}^n Q_i\right) \\ \text{s. t.} \quad & \begin{pmatrix} P_i & M_i \\ M_i^\top & Q_i \end{pmatrix} \succeq 0,\ i = 1, 2, \ldots, n \end{aligned} \tag{4.5}$$

where P_i and Q_i represent a particular factorization of matrix M_i.

To facilitate our analysis, the following theorem presents an alternative form of GLRAM in (4.1).

Theorem 4.2. *The optimal solution to (4.1) is equivalent to the following optimization problem*

$$\begin{aligned} \max_{\substack{Z_P \in \mathbf{S}_+^p \\ Z_Q \in \mathbf{S}_+^q}} \quad & \sum_{i=1}^n tr(M_i^\top Z_P M_i Z_Q) \\ s.\ t. \quad & 0 \preceq Z_P \preceq I_p, tr(Z_P) = k \\ & 0 \preceq Z_Q \preceq I_q, tr(Z_Q) = k \end{aligned} \tag{4.6}$$

Proof. As already shown in [172], GLRAM in (4.1) is equivalent to

$$\begin{aligned} \max_{\substack{L \in \mathbb{R}^{p\times k} \\ R \in \mathbb{R}^{q\times k}}} \quad & \sum_{i=1}^n \mathrm{tr}(M_i^\top L L^\top M_i R R^\top) \\ \text{s. t.} \quad & L^\top L = R^\top R = I_k \end{aligned}$$

By defining $Z_P = LL^\top$ and $Z_Q = RR^\top$, we have the problem in (4.6). $\square$

The following theorem reveals the relationship between GLRAM in (4.6) and the proposed formulation in (4.5)

Theorem 4.3. *Let the optimal value of (4.5) be denoted by f_1, and the optimal value of (4.6) be denoted by f_2. We have the following relationship between f_1 and f_2,*

$$f_2 \leq \frac{1}{2} f_1^2 \tag{4.7}$$

Proof. Let U_i and V_i be a factorization of matrix M_i, i.e., $M_i = U_i V_i^\top$. We can rewrite the objective function as

$$\sum_{i=1}^{n} \mathrm{tr}(U_i^\top Z_P U_i V_i^\top Z_Q V_i)$$

Note that both $U_i^\top Z_P U_i$ and $V_i^\top Z_Q V_i$ are positive semi-definite matrices and the above function is upper bounded as:

$$\begin{aligned}
& \sum_{i=1}^{n} \mathrm{tr}(U_i^\top Z_P U_i V_i^\top Z_Q V_i) \\
\leq & \sum_{i=1}^{n} \sqrt{\mathrm{tr}(U_i^\top Z_P U_i U_i^\top Z_P U_i)\mathrm{tr}(V_i^\top Z_Q V_i V_i^\top Z_Q V_i)} \\
\leq & \frac{1}{2} \sum_{i=1}^{n} \mathrm{tr}(U_i^\top Z_P U_i U_i^\top Z_P U_i) + \mathrm{tr}(V_i^\top Z_Q V_i V_i^\top Z_Q V_i) \\
\leq & \frac{1}{2} \sum_{i=1}^{n} [\mathrm{tr}(U_i^\top Z_P U_i)]^2 + [\mathrm{tr}(V_i^\top Z_Q V_i)]^2 \\
\leq & \frac{1}{2} \left(\sum_{i=1}^{n} \mathrm{tr}(U_i^\top Z_P U_i) + \mathrm{tr}(V_i^\top Z_Q V_i) \right)^2
\end{aligned}$$

The first step of the above derivation follows the Cauchy's inequality. We define $P_i = U_i U_i^\top$

and $Q_i = V_i V_i^\top$, and have the above inequality simplified as

$$\begin{aligned}
& \sum_{i=1}^{n} \mathrm{tr}(U_i^\top Z_P U_i V_i^\top Z_Q V_i) \\
\leq\ & \frac{1}{2}\left(\mathrm{tr}\left(Z_P\left[\sum_{i=1}^{n} P_i\right]\right) + \mathrm{tr}\left(Z_Q\left[\sum_{i=1}^{n} Q_i\right]\right)\right)^2 \\
\leq\ & \frac{1}{2}\left(S_k\left(\sum_{i=1}^{n} P_i\right) + S_k\left(\sum_{i=1}^{n} Q_i\right)\right)^2
\end{aligned}$$

Finally, using the property derived in [136], i.e.,

$$P = UU^\top, Q = VV^\top, X = UV^\top \Longleftrightarrow \begin{pmatrix} P & X \\ X^\top & Q \end{pmatrix} \succeq 0,$$

we have the result in the theorem. □

As revealed by Theorem 4.3, the optimal value of (4.5) provides an upper bound for the optimal value of (4.6), which to some degree justifies the usage of (4.5) for finding the optimal projection for matrices.

4.2.3 An SDP Formulation for SOPCA

Solving the optimization problem in (4.5) is difficult due to function $S_k(\cdot)$. The theorem below shows an SDP formulation that is equivalent to (4.5).

Theorem 4.4. *The problem in (4.5) is equivalent to the following optimization problem:*

$$\begin{aligned}
\max_{\substack{T_P \in \mathbf{S}_+^p \\ T_Q \in \mathbf{S}_+^q \\ Z_i \in \mathbb{R}^{p\times q}}} \quad & \sum_{i=1}^{n} tr(Z_i^\top M_i) \\
s.\ t. \quad & \begin{pmatrix} T_P & Z_i \\ Z_i^\top & T_Q \end{pmatrix} \succeq 0,\ i = 1, 2, \ldots, n \\
& T_P \preceq I,\ T_Q \preceq I \\
& tr(T_P) \leq k,\ tr(T_Q) \leq k
\end{aligned} \tag{4.8}$$

Proof. Similar to the analysis of Theorem 5.2, we first rewrite $S_k\left(\sum_{i=1}^n P_i\right)$ as

$$S_k\left(\sum_{i=1}^n P_i\right) = \max_{T_P \in \Delta_p(k)} \operatorname{tr}\left(T_P\left[\sum_{i=1}^n P_i\right]\right)$$

where $\Delta_p(k) = \{T \in \mathbb{R}_+^p : 0 \preceq T \preceq I_p, \operatorname{tr}(T) = k\}$. Hence, the optimization problem in (4.5) becomes

$$\begin{array}{ll} \min\limits_{\substack{P_i \in \mathbf{S}_+^p \\ Q_i \in \mathbf{S}_+^q}} \max\limits_{\substack{T_P \in \Delta_p(k) \\ T_Q \in \Delta_q(k)}} & \operatorname{tr}\left(\sum_{i=1}^n P_i T_P\right) + \operatorname{tr}\left(\sum_{i=1}^n Q_i T_Q\right) \\ \text{s. t.} & \begin{pmatrix} P_i & M_i \\ M_i^\top & Q_i \end{pmatrix} \succeq 0, i = 1, \ldots, n \end{array}$$

Since the above problem is a convex-concave optimization problem, we could switch minimization with maximization. In the next step, we will turn the minimization over P_i and Q_i into a maximization problem by deriving its dual form. In particular, we aim to derive the dual form for the following minimization problem

$$\begin{array}{ll} \min\limits_{P_i, Q_i} & \operatorname{tr}\left(\sum_{i=1}^n P_i T_P\right) + \operatorname{tr}\left(\sum_{i=1}^n Q_i T_Q\right) \\ \text{s. t.} & \begin{pmatrix} P_i & M_i \\ M_i^\top & Q_i \end{pmatrix} \succeq 0, i = 1, \ldots, n \end{array}$$

To this end, we construct the Lagrangian function $\mathcal{L}$

$$\begin{aligned} \mathcal{L} &= \operatorname{tr}\left(\sum_{i=1}^n P_i T_P\right) + \operatorname{tr}\left(\sum_{i=1}^n Q_i T_Q\right) \\ &\quad - \sum_{i=1}^n \Big(\operatorname{tr}(P_i A_i) + \operatorname{tr}(Q_i B_i) + 2\operatorname{tr}(Z_i M_i^\top)\Big) \end{aligned}$$

where

$$\begin{pmatrix} A_i & Z_i \\ Z_i^\top & B_i \end{pmatrix} \succeq 0$$

is the Lagrangian multiplier introduced for each LMI constraint. By setting the derivative of $\mathcal{L}$ with respect to P_i and Q_i to be zero, we have $T_P = A_i$ and $T_Q = B_i$. Hence, the resulting

dual problem becomes

$$\begin{aligned} \max_{Z_i} \quad & \sum_{i=1}^{n} \mathrm{tr}(Z_i M_i^\top) \\ \text{s. t.} \quad & \begin{pmatrix} T_P & Z_i \\ Z_i^\top & T_Q \end{pmatrix} \succeq 0, i = 1, \ldots, n \end{aligned}$$

Combining the above maximization problem with the maximization over T_P and T_Q, we have the theorem result. □

The formulation in (4.8) reveals an interesting aspect of the new formulation for SOPCA that is summarized by the following theorem.

Theorem 4.5. *The problem in (4.8) is equivalent to the following optimization problem if* $rank(T_P) = rank(T_Q) = k$

$$\begin{aligned} \max_{\substack{A \in \mathbb{R}^{p \times k} \\ B \in \mathbb{R}^{q \times k}}} \quad & \sum_{i=1}^{n} \|A^\top M_i B\|_{tr} \\ s.\ t. \quad & A^\top A = B^\top B = I_k \end{aligned} \tag{4.9}$$

where $T_P = AA^\top$ *and* $T_Q = BB^\top$.

Proof. First, since $\mathrm{rank}(T_P) = \mathrm{rank}(T_Q) = k$, we could rewrite T_P and T_Q as

$$T_P = \sum_{i=1}^{k} \lambda_i^P \mathbf{a}_i \mathbf{a}_i^\top = A \Sigma_P A^\top$$
$$T_Q = \sum_{i=1}^{k} \lambda_i^Q \mathbf{b}_i \mathbf{b}_i^\top = B \Sigma_Q B^\top$$

where $A = (\mathbf{a}_1, \ldots, \mathbf{a}_k)$ and $B = (\mathbf{b}_1, \ldots, \mathbf{b}_k)$ are eigenvectors of T_P and T_Q, and $\{\lambda_i^P\}_{i=1}^k$ and $\{\lambda_i^Q\}_{i=1}^k$ are the eigenvalues of T_P and T_Q, respectively. Since $0 \preceq T_P, T_Q \preceq I$, we have $0 \leq \lambda_i^P, \lambda_i^Q \leq 1$. Since the objective is to maximize $\mathrm{tr}(Z_i^\top M_i)$, and a larger T_P and T_Q will lead to a larger Z_i, we therefore have $\lambda_i^P = \lambda_i^Q = 1, i = 1, \ldots k$. Thus, T_P and T_Q can be simplified as $T_P = AA^\top$ and $T_Q = BB^\top$. Due to the LMI constraint, we have Z_i written as

$Z_i = AW_iB^\top$ where $W_i \in \mathbb{R}^{k\times k}$. Using W_i, we rewrite the LMI constraint as

$$\begin{pmatrix} I_k & W_i \\ W_i^\top & I_k \end{pmatrix} \succeq 0$$

The above constraint is also equivalent to $W_iW_i^\top \preceq I_k$. Hence, we could write W_i in its singular value decomposition form, i.e.,

$$W_i = \sum_{l=1}^{k} \gamma_l^i \mathbf{u}_l^i {\mathbf{v}_l^i}^\top$$

where $0 \leq \gamma_l^i \leq 1$ due to $W_iW_i^\top \preceq I_k$. As a result, $\mathrm{tr}(Z_i^\top M_i)$ is upper bounded by

$$\mathrm{tr}(Z_i^\top M_i) = \mathrm{tr}(W_iA^\top M_iB) \leq \|A^\top M_iB\|_{tr}$$

where the equality is taken when $\mathbf{u}_l^i$ and $\mathbf{v}_l^i$ are singular value decomposition of $A^\top M_iB$. We thus could replace the problem in (4.8) with the one that involves A and B, which leads to the theorem result. □

Using the above theorem result, it is easy to identify the relationship between (4.9) and (4.6). To make the comparison more obvious, we rewrite $\|A^\top M_iB\|_{tr}$ as

$$\|A^\top M_iB\|_{tr} = \mathrm{tr}\left(\sqrt{A^\top M_iBB^\top M_i^\top A}\right)$$

By letting $Z_P = AA^\top$ and $Z_Q = BB^\top$, we have the following relationship [136]:

$$\left(\mathrm{tr}(M_i^\top Z_P M_i Z_Q)\right)^{\frac{1}{2}} \leq \|A^\top M_iB\|_{tr}$$
$$r^{\frac{1}{2}}\left(\mathrm{tr}(M_i^\top Z_P M_i Z_Q)\right)^{\frac{1}{2}} \geq \|A^\top M_iB\|_{tr},$$

where r is the rank of $A^\top M_iB$, which clearly reveals the relationship between (4.6) and (4.9).

4.3 The Algorithm

Optimization problem involved in (4.8) is convex, thus the global optimal exists. We can resort to the standard SDP solving packages to solve (4.8). Directly solving (4.8) for large-

Dataset	Size	Dim	# of classes
USPS	4000	16 ×16=256	10
ORL	400	112×92=10304	40
AR	1638	400×350=140000	126

Table 4.1. Statistics of data sets

size data sets however, could be computationally expensive. Below, we give an alternative algorithm that obtains an approximate solution to (4.8).

Note that in (4.9), the trace norm $\sum_{i=1}^{n} \|A^\top M_i B\|_{tr}$ can be expressed as $\sum_{i=1}^{n} \mathrm{tr}\left(\sqrt{A^\top M_i^B [M_i^B]^\top A}\right)$, where $M_i^B = M_i B$. Using the following inequality:

$$\begin{aligned} & A^\top (M_i^B [M_i^B]^\top) A \\ = \; & A^\top (M_i^B [M_i^B]^\top)^{\frac{1}{2}} (M_i^B [M_i^B]^\top)^{\frac{1}{2}} A \\ \succeq \; & \left[A^\top (M_i^B [M_i^B]^\top)^{\frac{1}{2}} A\right] \left[A^\top (M_i^B [M_i^B]^\top)^{\frac{1}{2}} A\right], \end{aligned}$$

we have the following inequality holds:

$$\mathrm{tr}\left(\sqrt{A^\top (M_i^B [M_i^B]^\top) A}\right) \geq \mathrm{tr}\left(A^\top \sqrt{M_i^B [M_i^B]^\top} A\right)$$

Following the above inequality, we relax the problem in (4.9) as follows:

$$\begin{aligned} \max_{A \in \mathbb{R}^{p \times k}} \quad & \sum_{i=1}^{n} \mathrm{tr}\left(A^\top \sqrt{M_i^B [M_i^B]^\top} A\right) \\ \text{s. t.} \quad & A^\top A = I_k \end{aligned} \tag{4.10}$$

Clearly, the solution to (4.10) is the top k eigenvectors of $\sum_{i=1}^{n} \sqrt{M_i^B [M_i^B]^\top}$. Similarly, B can be computed from the top eigenvectors of matrix $\sum_{i=1}^{n} \sqrt{M_i^A [M_i^A]^\top}$, where $M_i^A = M_i^\top A$. Algorithm 1 summarizes the iterative algorithm.

4.4 Experiments

We evaluated the efficacy of the proposed method by using a task of image classification, and compared it to GLRAM and 2-D SVD, two state-of-the-art algorithms for SOPCA.

Algorithm 1 Second-Order PCA-Style Algorithm by Semi-Definite Programming

INPUT: a collection of matrices $\{M_i\}_{i=1}^n$, target dimension k
OUTPUT: left and right projection matrices A, B, and low dimensional representation $\{D_i\}_{i=1}^n$
Obtain an initial A_0 and set $i = 0$
while not converge **do**
  $i = i + 1$
  compute $M_R = \sum_{j=1}^n \sqrt{M_j^\top A_{i-1} A_{i-1}^\top M_j}$
  compute the top k eigenvectors $\{\mathbf{u}_l\}_{l=1}^k$ of M_R
  set $B_i = [\mathbf{u}_1, \cdots, \mathbf{u}_k]$
  compute $M_L = \sum_{j=1}^n \sqrt{M_j B_i B_i^\top M_j^\top}$
  compute the top k eigenvectors $\{\mathbf{v}_l\}_{l=1}^k$ of M_L
  set $A_i = [\mathbf{v}_1, \cdots, \mathbf{v}_k]$
end while
set $A = A_i$ and $B = B_i$
for $i = 1$ to n **do**
  $D_i = A^\top M_i B$
end for

More specifically, we represented each image using a matrix, and applied the algorithms to the training images to compute the left and right projection matrices L, R. Then the low dimensional representation A_i was computed for each test matrix X_i (i.e., image) as $A_i = L^\top X_i R$. Finally, we predicted class labels of test images using the 1-Nearest neighbor classifier in which the distance between two images, X_a and X_b, was computed by the Frobenius norm $\|A_a - A_b\|_F$, where A_a and A_b were the reduced representations of X_a and X_b. A 10-fold cross validation was used in the experiments, and the classification accuracy averaged over ten runs was reported. It is worthy to note that the purpose of designing this experiment is simply to demonstrate the advantages of the proposed convex formulation comparing to the non-convex formulations of GLRAM and 2-D SVD, better classification results can be achieved if more sophisticated processing procedure and state-of-the-art classifiers are applied.

Three image data sets were employed as the testing bed including ORL [133], AR [102], and USPS [64]. For ORL and USPS date sets, we used their original image size. For AR

data set, each image was cropped from row 100 to 500 and column 200 to 550 which was the same as [172]. The details of these data sets are summarized in Table 5.1.

As we design, we can see the difficulty of the task increases by two fold from the USPS data set to the AR data set. First, the dimensionality of the data in these data sets increases dramatically. For USPS, it is only 256, however in AR, it is $140,000$, more than 500 times higher than USPS. By switching to different data sets, we gradually vary the difficulty for dimensionality reduction, which can give us a clear picture about how these algorithms perform. Second, the ORL and AR data sets consist of human faces images which makes them a more challenging task for dimensionality reduction than the USPS digit images. Moreover, some faces in the AR data set vary with different facial expressions, illumination conditions, and occlusions (sun glasses and scarf). Combining high dimensionality and the large variance of face images, the AR data set is the most difficult one of the three data sets.

Table 4.2 shows the classification errors of the three algorithms with different k (i.e., the target dimension). First, we observe that all three methods have similar performance level on the USPS data set. We believe this may be due to the low dimensionality of the data points in USPS. Each image in USPS is only $16 * 16 = 256$ dimensional. As a result, even a very low dimensional representation is sufficient for capturing most of the information in the original data. In addition, the large number of samples in USPS make it an even easier task for dimensionality reduction. Second, for the ORL data set, all of the three methods still perform similarly except for $k = 2$, where the proposed algorithm yields a significantly better performance than the two baseline algorithms. Finally, for AR data set, the proposed method outperforms the two baseline methods significantly for almost all ks. We also observe that 2-D SVD performs significantly worse than the other two algorithms. In summary, we can conclude that the proposed algorithm is more effective than the two state-of-the-art algorithms for SOPCA, especially when the target dimension is very low.

4.5 Summary

In this chapter, we proposed aa convex formulation for the SOPCA problem. This contrasts to most existing algorithms for SOPCA which require solving non-convex optimization problems. We showed that the proposed formulation could be converted into a Semi-definite Programming Problem, we also presented an approximate algorithm to efficiently solve the related SDP problem and evaluated the proposed algorithm by image classification. Our experimental study shows the performance of the proposed algorithm is better than or comparable to GLRAM and 2-D SVD, two state-of-the-art algorithms for SOPCA.

	USPS data set								
	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$
Proposed	**0.3553**	0.0995	**0.0485**	**0.0385**	**0.0375**	**0.0377**	**0.0398**	**0.0405**	0.0428
	0.0001	0.0003	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
GLRAM	0.3565	**0.0978**	0.0505	**0.0385**	**0.0375**	0.0380	0.0403	0.0408	**0.0423**
	0.0002	0.0003	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
2-D SVD	0.3553	0.1108	0.0573	0.0428	0.0413	0.0380	0.0400	0.0408	0.0425
	0.0004	0.0002	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001

	ORL data set								
	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$
Proposed	**0.1725**	0.0550	**0.0275**	**0.0150**	0.0200	**0.0150**	**0.0150**	**0.0200**	**0.0175**
	0.0019	0.0012	0.0008	0.0007	0.0008	0.0004	0.0006	0.0005	0.0003
GLRAM	0.3000	0.0425	**0.0275**	0.0175	**0.0175**	0.0175	**0.0150**	**0.0200**	**0.0175**
	0.0036	0.0014	0.0008	0.0004	0.0006	0.0007	0.0006	0.0005	0.0003
2-D SVD	0.2225	**0.0400**	0.0300	0.0250	0.0200	0.0150	**0.0150**	**0.0200**	0.0200
	0.0041	0.0016	0.0007	0.0007	0.0005	0.0006	0.0006	0.0005	0.0005

	AR data set								
	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$
Proposed	**0.6590**	**0.4385**	**0.3128**	**0.4667**	**0.3974**	**0.3744**	**0.3333**	**0.3179**	**0.3077**
	0.0023	0.0095	0.0036	0.0031	0.0033	0.0034	0.0052	0.0049	0.0044
GLRAM	0.6797	0.6154	0.5128	0.4974	0.4538	0.4026	0.3667	0.3308	0.3154
	0.0041	0.0084	0.0044	0.0043	0.0038	0.0044	0.0034	0.0037	0.0030
2-D SVD	0.9282	0.9128	0.8718	0.8359	0.0877	0.8385	0.8231	0.8231	0.8256
	0.0018	0.0023	0.0037	0.0063	0.0054	0.0045	0.0065	0.0018	0.0028

Table 4.2. The classification errors of 1-NN classifier using the proposed dimensionality reduction algorithm for SOPCA, GLRAM and 2-D SVD. For each method, the first line is the mean error rates of the ten-fold cross validation and the second line is the standard deviations of the error rates. k represents the target dimension. The best performance of each case is highlighted by the bold font.

CHAPTER 5

Dimensionality Reduction for Functional Data Representation and Its Application to Large-Scale Image Retrieval by Local Features

In the previous chapter, we extended the data representation from vectors to matrices. In this chapter, we will study the problem of dimensionality reduction with even more challenging data representation, i.e., each datum is described as a different number of unordered vectors, which sometimes is referred to as the bag-of-features model. One way to deal with the bag-of-features representation is to associate with each datum a function from which the set of vectors are generated. The challenge of using functions to represent data is that the similarity between two data points is evaluated by the similarity between the associated functions, which is usually computationally expensive. It is therefore important to identify a low dimensional representation for those functions to speed up the computation of similarity between two functions and to facilitate the search of nearby neighbors, leading to the research of functional data dimensionality reduction. We study the problem of representing functions in a low dimensional space in the context of content-based image retrieval (CBIR) with images represented by the bag-of-features. In this chapter, we first briefly review the content-based image retrieval, and present how the problem of image matching can be cast into the problem of matching two different density functions. We then describe our approach

of functional data dimensionality reduction, which embeds each density function in a low dimensional space.

5.1 Content-Based Image Retrieval with Local Image Features

Content-based image retrieval (CBIR) is a long standing challenging problem in computer vision and multimedia. The earliest use of the term content-based image retrieval in the literature seems to have been by [60], to describe his experiments into automatic retrieval of images from a database by color and shape feature. The term has since been widely used to describe the process of retrieving desired images from a large collection on the basis of features that can be automatically extracted from the images themselves [40]. In the early ages, global features such as color [139, 109, 108], texture [22, 151, 92, 97, 96] and shape [112, 103] were widely used to represent images for CBIR. Recent studies [119, 145, 113, 89, 141, 75] have shown that local image features (e.g. SIFT descriptor [94]), often referred to as keypoints, are effective for identifying images with similar visual content. The key idea is to represent each image by a set of "interesting" patches extracted from the image. By representing every image patch with a high dimensional feature vector, each image is essentially represented by a bag of feature vectors, which is often referred to as the *bag-of-features* representation [29]. In the bag-of-features representation, the number of keypoints usually varies from image to image and the order among the keypoints is usually ignored in representing the visual content of images although they may be important. Given the bag-of-features representation, the similarity between two images are measured based on the"overlap" between the two sets of keypoints associated with the images. The challenge of applying the bag-of-features representation to image retrieval is how to efficiently measure the similarity between two unordered set of vectors with different number of vectors in each set.

A straightforward method to compute the similarity between two sets of keypoints is the

best partial matching which finds the best mapping between the keypoints in the two images that has the overall shortest distance [94, 104]. The main shortcoming of the optimal partial matching is its high computational cost since we need to compute the distance between every possible pair of keypoints of the two images. Several methods have been proposed to improve the computational efficiency of the optimal partial matching [146, 52, 79, 106]. The central ideal of these methods is that instead of using the bag-of-features representation directly, they manage to convert the bag-of-features into certain consistent form and then standard techniques for similarity measurement can be easily applied. For example, in [146] and [52], the keypoints are quantized into "visual words" or "multi-resolution bins" resulting a histogram or multi-resolution histogram representation of the images. These methods, however, suffer from the information loss during the quantization procedure [120] which often leads to suboptimal results. One possible approach to alleviate the problem mentioned above is to assume that the observed feature vectors are sampled from an unknown distribution. This assumption leads to a natural representation that represents each image by a density function of keypoints [79, 106, 83, 30]. The similarity between two images is then computed by the distance between two density functions derived from the observed keypionts. More specifically, let $\mathcal{G} = \{\mathcal{I}_1, \ldots, \mathcal{I}_C\}$ be the collection of C images, and each image $\mathcal{I}_i$ be represented by a set of n_i keypoints $\{\mathbf{x}_1^i, \ldots, \mathbf{x}_{n_i}^i\}$, where each key point $\mathbf{x}_i \in \mathbb{R}^d$ is a d dimensional vector. We assume that the keypoints of image $\mathcal{I}_i$ are randomly sampled from an unknown distribution $p(\mathbf{x}|\mathcal{I}_i)$. Furthermore, we can assume the distribution of $p(\mathbf{x}|\mathcal{I}_i)$ has a parametric form, for example, Gaussian Mixture Model (GMM) [106]. The parameter of the model can be estimated by maximizing the log likelihood of the observed keypoints of the image. After obtaining the generative models $p(\mathbf{x}|\mathcal{I}_i)$ of the images, the similarity between images $\mathcal{I}_i$ and $\mathcal{I}_j$ is computed as the model distance between $p(\mathbf{x}|\mathcal{I}_i)$ and $p(\mathbf{x}|\mathcal{I}_j)$. One of the most widely used methods is the Kullback-Leibler divergence [106]:

$$D(p(\mathbf{x}|\mathcal{I}_i), p(\mathbf{x}|\mathcal{I}_j)) \quad = \quad \int p(\mathbf{x}|\mathcal{I}_i) \log \frac{p(\mathbf{x}|\mathcal{I}_i)}{p(\mathbf{x}|\mathcal{I}_j)} d\mathbf{x} \tag{5.1}$$

The disadvantage of fitting keypoints by a parametric model is that the resulting model

could be significantly constrained by the family of parametric models, and consequently leads to a suboptimal performance for image retrieval. Besides the parametric models, we can also use nonparametric methods, for instance, kernel density estimation [115] to estimate the underlying density function $p(\mathbf{x}|\mathcal{I})$, i.e.,

$$p(\mathbf{x}|\mathcal{I}) = \frac{1}{n}\sum_{i=1}^{n}\kappa(\mathbf{x}, \mathbf{x}_i) \tag{5.2}$$

where $\kappa(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}_+$ is the kernel density function that is normalized as $\int d\mathbf{z}\kappa(\mathbf{x}, \mathbf{z}) = 1$. Given the density function in (5.2), we follow the work of statistical language models [122] for document retrieval, and measure the similarity of $\mathcal{I}$ to the query image $\mathcal{Q}$ by the logarithm of the query likelihood $p(\mathcal{Q}|\mathcal{I})$, i.e.,

$$\log p(\mathcal{Q}|\mathcal{I}) = \sum_{i=1}^{m}\log p(\mathbf{q}_i|\mathcal{I}) = \sum_{i=1}^{m}\log\left(\frac{1}{n}\sum_{j=1}^{n}\kappa(\mathbf{x}_j, \mathbf{q}_i)\right)$$

Although the idea of modeling a bag-of-features by a density function and model query-to-image similarity by a log-likelihood have been studied by multiple authors (e.g., [166, 76, 86, 107, 79]), there are one computational challenge that makes this approach difficult to scale to image retrieval problems with large databases, i.e., an naive implementation of this approach requires a linear scan of the entire database before finding the images with the largest similarity. In this chapter, we will address this challenge by developing dimensionality reduction methods for functional data representation.

5.2 Dimensionality for Functional Data Representation

In order to make an efficient similarity measurement between functions, we consider an alternative approach of estimating the density function for image $\mathcal{I}$. We assume that for any image $\mathcal{I}$ in the gallery $\mathcal{G}$, its density function $p(\mathbf{x}|\mathcal{I})$ is expressed as a weighted mixture

models:

$$p(\mathbf{x}|\mathcal{I}) = \sum_{i=1}^{N} \alpha_i \kappa(\mathbf{x}, \mathbf{c}_i) \tag{5.3}$$

where $\mathbf{c}_i \in \mathbb{R}^d, i = 1, \ldots, N$ is a collection of N points (centers) that are randomly selected from all the keypoints observed in $\mathcal{G}$ and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N)$ is a probability distribution used to combine different kernel functions.

The choice of randomly selected centers, although may seem to be naive at the first glance, is in fact strongly supported by the consistency results of kernel density estimation [115]. In particular, the kernel density function constructed by randomly selected centers is almost "optimal" when the number of centers is very large. The number of centers N is usually chosen to be very large, in order to cover the diverse visual content of images.

It is important to note that unlike (5.2), the weights $\boldsymbol{\alpha}$ in (5.3) are unknown and need to be determined for each image. As will be shown later, with an appropriate choice of kernel function $\kappa(\cdot, \cdot)$, the resulting weights $\boldsymbol{\alpha}$ will be sparse with most of the elements being zero. This is ensured by the fact that in a high dimensional space, almost any two randomly selected data points are far away from each other. It is the sparsity of $\boldsymbol{\alpha}$ that makes it possible to efficiently compute the similarity between functions and identify images that are visually similar to the query without having to scan the entire image database.

In order to use the density function in (5.3), we need to efficiently estimate the combination weights $\boldsymbol{\alpha}$. By assuming keypoints $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are randomly sampled from $p(\mathbf{x}|\mathcal{I})$, our first attempt is to estimate $\boldsymbol{\alpha}$ by a maximum likelihood estimation, i.e.,

$$\boldsymbol{\alpha} = \arg\max_{\boldsymbol{\alpha} \in \Delta} \mathcal{L}(\mathcal{I}, \boldsymbol{\alpha}) = \sum_{i=1}^{n} \log \left(\sum_{j=1}^{N} \alpha_j \kappa(\mathbf{x}_i, \mathbf{c}_j) \right) \tag{5.4}$$

where $\Delta = \{\boldsymbol{\alpha} \in [0, 1]^C : \sum_{i=1}^{C} \alpha_i = 1\}$ defines a simplex of probability distributions. It is easy to verify that the problem in (5.4) is convex and has a global optimal solution.

Although we can directly apply the standard optimization approaches to find the optimal solution $\boldsymbol{\alpha}$ for (5.4), it is in general computationally expensive because

- We have to solve (5.4) for every image. Even if the optimization algorithm is efficient and can solve the problem within one second, for a database with a million of images, it will take more than 277 hours to complete the computation.
- The number of weights α to be determined is very large. To achieve the desired performance of image retrieval, we often need a very large number of centers, for example one million. As a result, it requires solving an optimization problem with million variables even for a single optimization problem in (5.4).

In order to address the computational challenge, we choose the following local kernel function for this study

$$\kappa(\mathbf{x}, \mathbf{c}) \propto I(|\mathbf{x} - \mathbf{c}|_2 \leq \rho) \tag{5.5}$$

where $I(z)$ is an indicator function that outputs 1 if z is true and zero otherwise. The parameter $\rho > 0$ is a predefined constant that defines the locality of the kernel function and its value is determined empirically. The proposition below shows the sparsity of the solution $\boldsymbol{\alpha}$ for (5.4).

Proposition 1. *Given the local kernel function defined in (5.5), for the optimal solution α to (5.4), we have $\alpha_j = 0$ for center $\mathbf{c}_j$ if $\max\limits_{1 \leq i \leq n} |\mathbf{c}_j - \mathbf{x}_i|_2 > \rho$*

Proposition 1 follows directly from the fact that $\kappa(\mathbf{c}_j, \mathbf{x}_i) = 0, i = 1, \ldots, n$ if $\max_{1 \leq i \leq n} |\mathbf{c}_j - \mathbf{x}_i|_2 > \rho$. As implied by Proposition 1, α_j will be nonzero only if the center $\mathbf{c}_j$ is within a distance ρ of some keypoints. By setting ρ to a small value, we will only have a small number of non-zero α_j. We can quickly identify the subset of centers with non-zero α_j by an efficient range search, for example using k-d tree [93]. In our study, this step reduces the number of variables from 1 million to about $1,000$.

Although Proposition 1 allows us to reduce the number of variables dramatically, we still have to find a way to solve (5.4) efficiently. To this end, we resort to the bound optimization strategy that leads to a simple iterative algorithm for optimizing (5.4): we denote by $\boldsymbol{\alpha}'$ the current solution and by $\boldsymbol{\alpha}$ the updated solution for (5.4). It is straightforward to show that

$\{\mathcal{L}(\mathcal{I}, \boldsymbol{\alpha}) - \mathcal{L}(\mathcal{I}, \boldsymbol{\alpha}')\}$ is bounded as follows

$$\begin{aligned}\mathcal{L}(\mathcal{I}, \boldsymbol{\alpha}) - \mathcal{L}(\mathcal{I}, \boldsymbol{\alpha}') &= \sum_{i=1}^{n} \log \frac{\sum_{j=1}^{N} \alpha_j \kappa(\mathbf{x}_i, \mathbf{c}_j)}{\sum_{j=1}^{N} \alpha'_j \kappa(\mathbf{x}_i, \mathbf{c}_j)} \\ &\geq \sum_{i=1}^{n} \sum_{j=1}^{N} \frac{\alpha'_j \kappa(\mathbf{x}_i, \mathbf{c}_j)}{\sum_{l=1}^{N} \alpha'_j \kappa(\mathbf{x}_i, \mathbf{c}_l)} \log \frac{\alpha_j}{\alpha'_j} \end{aligned} \tag{5.6}$$

By maximizing the lower bound in (5.6), we have the following updating rule for α

$$\alpha_j = \frac{1}{Z} \sum_{i=1}^{n} \frac{\alpha'_j \kappa(\mathbf{x}_i, \mathbf{c}_j)}{\sum_{l=1}^{N} \alpha'_l \kappa(\mathbf{x}_i, \mathbf{c}_l)} \tag{5.7}$$

where Z is the normalization factor ensuring $\sum_{j=1}^{N} \alpha_j = 1$. Note that $\boldsymbol{\alpha}$ obtained by iteratively running the updating equation in (5.7) is indeed globally optimal because the optimization problem in (5.4) is convex.

We can further simplify the computation of $\boldsymbol{\alpha}$ as: we first initialize $\alpha_j = 1/N, i = 1, \ldots, N$, and then obtain the solution $\boldsymbol{\alpha}$ by only running the iteration once, i.e.,

$$\alpha_j = \frac{1}{n} \sum_{i=1}^{n} \frac{\kappa(\mathbf{x}_i, \mathbf{c}_j)}{\sum_{l=1}^{N} \kappa(\mathbf{x}_i, \mathbf{c}_l)} \tag{5.8}$$

We emphasize that although the solution in (5.8) is approximated in only one update, it is however the exact optimal solution when the keypoints $\{\mathbf{x}_i\}_{i=1}^{N}$ are far apart from each other, as shown by the following theorem.

Theorem 5.1. *Let the kernel function be (5.5). Assume that all the keypoints $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are separated by at least 2ρ. The solution α in (5.8) optimizes the problem in (5.4).*

Proof. When any two keypoints $\mathbf{x}_i$ and $\mathbf{x}_j$ are separated by at least 2ρ, we have $\kappa(\mathbf{x}_i, \mathbf{c}_k)\kappa(\mathbf{x}_j, \mathbf{c}_k) = 0$ for any center $\mathbf{c}_k$. This implies that no key point could make contribution to the estimation of weight α_k simultaneously for two different centers in 5.7 . As a

result, the expression in 5.7 could be rewritten as

$$\begin{aligned}\alpha_j &= \frac{1}{Z}\sum_{i=1}^{n} I(|\mathbf{x}_i - \mathbf{c}_j| \leq \rho)\frac{\alpha'_j}{\sum_{l=1}^{N}\alpha'_l\kappa(\mathbf{x}_i, \mathbf{c}_l)} \\ &= \frac{1}{Z}\sum_{i=1}^{n} I(|\mathbf{x}_i - \mathbf{c}_j| \leq \rho)\frac{\alpha'_j}{\alpha'_j\kappa(\mathbf{x}_i, \mathbf{c}_j)} \\ &= \frac{1}{Z}\sum_{i=1}^{n} I(|\mathbf{x}_i - \mathbf{c}_j| \leq \rho)\end{aligned}$$

As a result, the updating equation will give the fixed solution, which is the global optimal solution. □

In algorithm 2, we summarize the procedure of computing $\boldsymbol{\alpha}_i$ for each image $\mathcal{I}_i$ in the image collection. The key step of computing each $\boldsymbol{\alpha}_i$ is how to efficiently compute the value of kernel function in (5.5). In the algorithm, we resort to the k-d tree based range search to achieve the goal. More specifically, we first build a k-d tree for all the key points of images in the collection. For each center, we then search the keypints which are within the distance ρ of that center using the k-d tree. For all the keypoints that are within the distance ρ of that center, their values to the kernel function (5.5) for this center is 1 and for other keypoints the value is 0. After we conduct the range search for every centers, we obtain the value of (5.5) for every pair of keypoints and centers for all the image in the collection which can be used directly for computing $\boldsymbol{\alpha}_i$ for each image.

Regularization Although the sparse solution resulting from the local kernel is computationally efficient, the sparse solution may lead to a poor estimation of query-likelihood, as demonstrated in the study of statistical language model [99]. To address this challenge, we introduce $\boldsymbol{\alpha}^g = (\alpha^g_1, \ldots, \alpha^g_N)$, a global set of weights used for kernel density function. $\boldsymbol{\alpha}^g$ plays the same role as the background langauge model in statistical language models [99]. We defer the discussion of how to compute α^g to the end of this section. Given the global set of weights $\boldsymbol{\alpha}^g$, we introduce $\mathrm{KL}(\boldsymbol{\alpha}^g \| \boldsymbol{\alpha})$, the Kullback-Leibler divergence [106] between

Algorithm 2 Compute weight vector $\boldsymbol{\alpha}_i$ of each image $\mathcal{I}_i$ in the image collection

1: INPUT:

- Image collection $\mathcal{G} = \{\mathcal{I}_1, \ldots, \mathcal{I}_C\}$ with each image $\mathcal{I}_i$ represented by a set of keypoints
- Number of random centers N
- Distance threshold ρ

2: Randomly select N keypoints as the centers $\mathbf{z}_1, \ldots, \mathbf{z}_N$ from X that consists of all the keypoints detected from images in $\mathcal{G}$
3: Construct a randomized k-d tree T for all the keypoints in X
4: **for** $l = 1$ to N **do**
5: Using k-d tree T, search keypoints which are within the distance ρ of the center $\mathbf{z}_l$.
6: For all the returned keypoints, set their values of the kernel function (5.5) to be 1. For all other keypoints, set the value to be 0.
7: **end for**
8: **for** $i = 1$ to C **do**
9: Compute each element in the weight vector $\boldsymbol{\alpha}_i$ of image $\mathcal{I}_i$ using (5.8)
10: **end for**

$\boldsymbol{\alpha}^g$ and $\boldsymbol{\alpha}$, as a regularizer in (5.4), i.e.,

$$\boldsymbol{\alpha} = \arg\max_{\boldsymbol{\alpha} \in \Delta} \mathcal{L}(\mathcal{I}, \boldsymbol{\alpha}) - \lambda \mathrm{KL}(\boldsymbol{\alpha}^g \| \boldsymbol{\alpha}) \tag{5.9}$$

where $\lambda > 0$ is introduced to weight the importance of the regularizer. As indicated in (5.9), by introducing the KL divergence as the regularizer, we prefer the solution $\boldsymbol{\alpha}$ that is similar to $\boldsymbol{\alpha}^g$. Note that (5.9) is equivalent to the MAP estimation of $\boldsymbol{\alpha}$ by introducing a Dirichlet prior $\mathrm{Dir}(\boldsymbol{\alpha}) \propto \prod_{i=1}^{N} [\alpha_i]^{\beta_i}$, where $\beta_i = \lambda \alpha_i^g$. Similar to the bound optimization strategy used for solving (5.4), we have the following approximate solution for (5.9)

$$\alpha_j = \frac{1}{n + \lambda} \left(\lambda \alpha_j^g + \sum_{i=1}^{n} \frac{\kappa(\mathbf{x}_i, \mathbf{c}_j)}{\sum_{l=1}^{N} \kappa(\mathbf{x}_i, \mathbf{c}_j)} \right) \tag{5.10}$$

It is important to note that, according to (5.10), the solution for $\boldsymbol{\alpha}$ is no longer sparse if $\boldsymbol{\alpha}^g$ is not sparse, which could potentially lead to a high computational cost in image matching. We will discuss a method in the next section that explicitly addresses this computational challenge.

The remaining question is how to estimate $\boldsymbol{\alpha}^g$, the global set of weights. To this end, we search for the weight $\boldsymbol{\alpha}^g$ that can explain all the keypoints observed in all the images of gallery $\mathcal{G}$, i.e.,

$$\boldsymbol{\alpha}^g = \arg\max_{\boldsymbol{\alpha}^g \in \Delta} \sum_{i=1}^{C} \mathcal{L}(\mathcal{I}_i, \boldsymbol{\alpha}^g) \tag{5.11}$$

Although we can employ the same bound optimization strategy to estimate $\boldsymbol{\alpha}^g$, we describe below a simple approach that directly utilizes the solution $\boldsymbol{\alpha}$ for individual images to construct $\boldsymbol{\alpha}^g$. We denote by $\boldsymbol{\alpha}^i = (\alpha_1^i, \ldots, \alpha_N^i)$ the optimal solution that is obtained by maximizing the log-likelihood $\mathcal{L}(\mathcal{I}_i, \boldsymbol{\alpha}^i)$ of the keypoints observed in image $\mathcal{I}_i$. Given $\boldsymbol{\alpha}^i$ that maximizes $\mathcal{L}(\mathcal{I}_i, \boldsymbol{\alpha}^i)$, we have

$$\mathcal{L}(\mathcal{I}_i, \boldsymbol{\alpha}^g) \approx \mathcal{L}(\mathcal{I}_i, \boldsymbol{\alpha}^i) + \frac{1}{2}(\boldsymbol{\alpha}^g – \boldsymbol{\alpha}^i)^\top \nabla^2 \mathcal{L}(I_i, \boldsymbol{\alpha}^i)(\boldsymbol{\alpha}^g – \boldsymbol{\alpha}^i) \tag{5.12}$$

S Hessian matrix $\nabla^2 \mathcal{L}(\mathcal{I}_i, \boldsymbol{\alpha})$ is computed as $\nabla^2 \mathcal{L}(\mathcal{I}_i, \boldsymbol{\alpha}) = -\sum_{k=1}^{n_i} \mathbf{u}_i^k [\mathbf{u}_i^k]^\top$, where $\mathbf{u}_i^k \in \mathbb{R}^N$ is a vector defined as $[\mathbf{u}_i^k]_j = \kappa(\mathbf{x}_k^i, \mathbf{c}_j)/(\sum_{l=1}^{N} \alpha_j \kappa(\mathbf{x}_k^i, \mathbf{c}_j))$. The lemma below allows us to bound the Hessian matrix $\nabla^2 \mathcal{L}(\mathcal{I}_i, \boldsymbol{\alpha}^i)$.

Lemma 1. *$NI \succeq -\nabla^2 \mathcal{L}(\mathcal{I}_i, \boldsymbol{\alpha}^i)$.*

Proof. To bound the maximum eigenvalue $-\nabla^2 \mathcal{L}(\mathcal{I}_i, \boldsymbol{\alpha}^i)$, we consider the quantity $\gamma^\top \nabla^2 \mathcal{L}(\mathcal{I}_i, \boldsymbol{\alpha}^i)\gamma$ with $|\gamma|_2 = 1$.

$$\begin{aligned}
\gamma^\top \nabla^2 \mathcal{L}(\mathcal{I}_i, \boldsymbol{\alpha}^i)\gamma &= \sum_{k=1}^{n_i} \frac{[\sum_{j=1}^{N} \gamma_j \kappa(\mathbf{x}_k^i, \mathbf{c}_j)]^2}{[\sum_{j=1}^{N} \alpha_j \kappa(\mathbf{x}_k^i, \mathbf{c}_j)]^2} \\
&\leq \left(\sum_{k=1}^{n_i} \frac{\sum_{j=1}^{N} |\gamma_j| \kappa(\mathbf{x}_k^i, \mathbf{c}_j)}{\sum_{j=1}^{N} \alpha_j \kappa(\mathbf{x}_k^i, \mathbf{c}_j)} \right)^2
\end{aligned}$$

Define $\eta_j = |\gamma_j|/(\sum_{j=1}^{N} |\gamma_j|)$ and $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_N)$. Define $t = \sum_{j=1}^{N} |\gamma_j|$. We have

$$\gamma^\top \nabla^2 \mathcal{L}(\mathcal{I}_i, \boldsymbol{\alpha}^i)\gamma \;\; \leq \;\; t^2 \left(\sum_{k=1}^{n_i} \frac{\sum_{j=1}^{N} \eta_j \kappa(\mathbf{x}_k^i, \mathbf{c}_j)}{\sum_{j=1}^{N} \alpha_j \kappa(\mathbf{x}_k^i, \mathbf{c}_j)} \right)$$

Since $\boldsymbol{\alpha}^i$ maximizes $\mathcal{L}(\mathcal{I}_i, \boldsymbol{\alpha})$, we have

$$(\boldsymbol{\eta} - \boldsymbol{\alpha}^i)^\top \nabla \mathcal{L}(\mathcal{I}_i, \boldsymbol{\alpha}) \leq 0,$$

which implies

$$\sum_{k=1}^{n_i} \frac{\sum_{j=1}^{N} \eta_j \kappa(\mathbf{x}_k^i, \mathbf{c}_j)}{\sum_{j=1}^{N} \alpha_j \kappa(\mathbf{x}_k^i, \mathbf{c}_j)} \leq 1$$

Since $t \leq \sqrt{N}$, we have $\nabla^2 \mathcal{L}(\mathcal{I}_i, \boldsymbol{\alpha}^i) \succeq -NI$. □

Using the result in Lemma 5.1, the objective function in (5.11) can be approximated as

$$\sum_{i=1}^{C} \mathcal{L}(\mathcal{I}_i, \boldsymbol{\alpha}^g) \approx \sum_{i=1}^{C} \mathcal{L}(\mathcal{I}_i, \boldsymbol{\alpha}^i) - \frac{N}{2} \sum_{i=1}^{C} |\boldsymbol{\alpha}^i - \boldsymbol{\alpha}^g|_2^2 \tag{5.13}$$

The global weights $\boldsymbol{\alpha}^g$ maximizing (5.13) is $\boldsymbol{\alpha}^g = \frac{1}{C} \sum_{i=1}^{C} \boldsymbol{\alpha}^i$ which shows that $\boldsymbol{\alpha}^g$ can be computed as an average of $\{\boldsymbol{\alpha}^i\}_{i=1}^{C}$ that are optimized for individual images.

5.3 Efficient Image Search

Given the kernel density function $p(\mathbf{x}|\mathcal{I}_i)$ for each image in gallery $\mathcal{G}$ and a query $\mathcal{Q}$, the next question is how to efficiently identify the subset of images that are likely to be visually similar to the query $\mathcal{Q}$ and furthermore rank those images in the descending order of their similarity. Following the framework of statistical language models for text retrieval, we estimate the similarity by the likelihood of generating the keypoints $\{\mathbf{q}_i\}_{i=1}^{m}$ observed in the query $\mathcal{Q}$, i.e.,

$$\log p(\mathcal{Q}|\mathcal{I}_i) = \sum_{k=1}^{m} \log \left(\sum_{j=1}^{N} \alpha_j^i \kappa(\mathbf{q}_k, \mathbf{c}_j) \right) \tag{5.14}$$

where $\boldsymbol{\alpha}^i = (\alpha_1^i, \ldots, \alpha_N^i)$ are the weights for constructing the kernel density function for image $\mathcal{I}_i$. Clearly, a naive implementation will require a linear scan of all the images in the database before the subset of similar ones is found. To achieve the efficient image retrieval, we need to exploit the sparse structure of α in (5.10). We define

$$\widehat{\alpha}_j^i = \frac{1}{n_i} \sum_{k=1}^{n_i} \frac{\kappa(\mathbf{x}_k^i, \mathbf{c}_j)}{\sum_{l=1}^{N} \kappa(\mathbf{x}_k^i, \mathbf{c}_l)} \tag{5.15}$$

We then write α_j^i as

$$\alpha_j^i = \frac{\lambda}{n_i + \lambda}\alpha_j^g + \frac{n_i}{n_i + \lambda}\widehat{\alpha}_j^i \tag{5.16}$$

Note that although $\widehat{\alpha}_j^i$ is sparse, α_j^i is not. Our goal is to effectively explore the sparsity of $\widehat{\alpha}_j^i$ for efficient image retrieval. Using the expression in (5.16), we have $\log p(\mathcal{Q}|\mathcal{I}_i)$ expressed as

$$\begin{aligned}\log p(\mathcal{Q}|\mathcal{I}_i) &= \sum_{j=1}^{m} \log\left(\sum_{l=1}^{N}\left(\frac{\lambda}{n_i+\lambda}\alpha_l^g + \frac{n_i}{n_i+\lambda}\widehat{\alpha}_l^i\right)\kappa(\mathbf{x}_j, \mathbf{c}_l)\right) \\ &= \sum_{j=1}^{m} \log\left(1 + \frac{n_i}{\lambda}\frac{\sum_{l=1}^{N}\widehat{\alpha}_l^i\kappa(\mathbf{x}_j,\mathbf{c}_l)}{\sum_{l=1}^{N}\alpha_l^g\kappa(\mathbf{x}_j,\mathbf{c}_l)}\right) + s_Q\end{aligned} \tag{5.17}$$

where

$$s_Q = \sum_{j=1}^{m}\log\left(\frac{\lambda}{n_i+\lambda}\right) + \sum_{j=1}^{m}\log\left(\sum_{l=1}^{N}\alpha_l^g\kappa(\mathbf{x}_j,\mathbf{c}_l)\right) \tag{5.18}$$

Note that (i) the second term of s_Q is independent of the individual images for the same query, and (ii) $\log p(\mathcal{Q}|\mathcal{I}_i) \geq s_Q$ for any image $\mathcal{I}_i$. Given the above facts, our goal is to efficiently find the subset of images whose query log-likelihood is *strictly* larger than s_Q, i.e., $\log p(\mathcal{Q}|\mathcal{I}_i) > s_Q$. To this end, we consider the following procedure:

- *Finding the relevant centers $\mathcal{C}_Q$ for a given query $\mathcal{Q}$.* Given a query image $\mathcal{Q}$ with keypoints $\mathbf{q}_1, \ldots, \mathbf{q}_m$, we first identify the subset of centers, denoted by $\mathcal{C}_Q$, that are within distance ρ of the keypoints in $\mathcal{Q}$, i.e., $\mathcal{C}_Q = \{\mathbf{c}_j : \exists \mathbf{q}_k \in \mathcal{Q} \text{ s. t. } |\mathbf{q}_k - \mathbf{c}_j|_2 \leq \rho\}$.
- *Finding the candidates of similar images using the relevant centers.* Given the relevant centers in $\mathcal{C}_Q$, we find the subset of images that have at least one non-zero $\widehat{\alpha}_j^i$ for the centers in $\mathcal{C}_Q$, i.e.,

$$\mathcal{R}_Q = \left\{\mathcal{I}_i \in \mathcal{G} : \sum_{\mathbf{c}_j \in \mathcal{C}_Q}\widehat{\alpha}_j^i > 0\right\} \tag{5.19}$$

Theorem 5.2 shows that all the images with query log-likelihood larger than s_Q belong to $\mathcal{R}_Q$.

Theorem 5.2. *Let $\mathcal{S}_Q$ denote the set of images with query log-likelihood larger than s_Q, i.e., $\mathcal{S}_Q = \{\mathcal{I}_i \in \mathcal{G} : \log p(\mathcal{Q}|\mathcal{I}_i) > s_Q\}$. We have $\mathcal{S}_Q = \mathcal{R}_Q$.*

It is easy to verify the above theorem. In order to efficiently construct $\mathcal{R}_Q$ (or $\mathcal{S}_Q$) for a given query $\mathcal{Q}$, we exploit the technique of invert indexing [99]: we preprocess the images to obtain a list for each $\mathbf{c}_j$, denoted $\mathcal{V}_j$, that includes all the images $\mathcal{I}_i$ with $\widehat{\alpha}_j^i > 0$. Clearly, we have

$$\mathcal{R}_Q = \bigcup_{\mathbf{c}_j \in \mathcal{C}_Q} \mathcal{V}_j \tag{5.20}$$

Algorithm 3 summarizes the procedure of efficient image retrieval.

5.4 Comparing to The-State-Of-The-Art Methods

Bag-of-words model [146] is one the most successful and widely used methods for large scale image retrieval. Motivated by the success in text information retrieval [132], the bag-of-words model first quantizes image features to a vocabulary of "visual words", and represents each image by the counts of visual words or a histogram. Standard text retrieval techniques can then be applied to identify the images that share similar visual content as the query image. The quantization is typically achieved by grouping all the keypoints into a specified number of clusters using a clustering algorithm. A number of studies have shown promising performance of the bag-of-words approach for image/object retrieval [119, 145, 113, 89, 143, 141, 75]

To better understand the proposed method in (5.3), we compare it to the bag-of-words model. More specifically, we can view each random center $\mathbf{c}_i$ as a different visual word and each $\boldsymbol{\alpha}$ as a histogram vector. One computational advantage of the proposed method is that, while the bag-of-words model requires clustering all the keypoints into a large number of clusters, the proposed method only needs to randomly select a number of points from the date which is computationally efficient. Although recent progress on approximate nearest neighbor search [93, 31, 89, 144, 110] has made it feasible to group billions of keypoints into

Algorithm 3 Efficient image retrieval algorithm

1: INPUT:

- A query image Q with keypoints $\mathbf{q}_1, \ldots, \mathbf{q}_m$
- Inverted indices $\mathcal{V}_j, j = 1, \ldots, N$
- Number of images to be retrieved, k

2: OUTPUT:

- k images sorted descendingly by their similarity to the query image

3: Construct $\mathcal{C}_Q$ of the query Q as

$$\mathcal{C}_Q = \left\{ \mathbf{c}_j : \exists \mathbf{q}_k \in \mathcal{Q} \text{ s. t. } |\mathbf{q}_k - \mathbf{c}_j|_2 \leq \rho \right\}$$

4: Construct candidate image set $\mathcal{R}_Q$ of query Q as

$$\mathcal{R}_Q = \bigcup_{\mathbf{c}_j \in \mathcal{C}_Q} \mathcal{V}_j$$

5: **for** every image $\mathcal{I}_i$ in $\mathcal{R}_Q$ **do**
6: Compute the likelihood $\log p(\mathcal{Q}|\mathcal{I}_i)$ using (5.17)
7: **end for**
8: Sort images in $\mathcal{R}_Q$ based on their likelihood $\log p(\mathcal{Q}|\mathcal{I}_i)$
9: Return the first k images in $\mathcal{R}_Q$

millions of clusters, the computational cost is still very high. We will see this clearly later in our empirical study.

Second, in the bag-of-words model, we need to map each keypoint to the closest visual word(s). Since the computational cost of this procedure is linear in the number of keypoints, it is time consuming when the number of keypoints is very large; The proposed method, however, only needs to conduct a range search for every randomly selected centers which is in general significantly smaller than the number of key points, for example, one million centers v.s. on billion keypoints. This computational saving makes the proposed method more suitable for large image databases than the bag-of-words model.

Third, in the bag-of-words model, the radius of clusters (i.e., the maximum distance between the keypoints in a cluster and its center) could vary significantly from cluster to cluster. As a result, for cluster with large radius, two keypoints can be mapped to the same visual word even if they differ significantly in visual features, leading to an inconsistent criterion for keypoints quantization and potentially suboptimal performance in retrieval; On the contrary, the proposed method uses a range search for each center which ensures that only "similar" keypoints, which are within the distance of r to the center, will contribute to the corresponding element in the weight $\boldsymbol{\alpha}$ of that center.

Lastly, a keypoint is ignored by the proposed method if its distances to all the centers are larger than the threshold. The underlying rationale is that if a keypoint is far away from all centers, it is very likely to be an outlier and therefore should be ignored; While in the bag-of-words model, every keypoint must be mapped to a cluster center even if the keypoint is far away from all the cluster centers. We will see this advantage of the proposed method clearly demonstrated in the experiments.

5.5 Experiments

5.5.1 Datasets

To evaluate the proposed method for large-scale image search, we conduct experiments on three benchmark data sets: (1) tattoo image dataset (**tattoo**) with about $100,000$ images. (2) Oxford building dataset with $5,000$ images (**Oxford5K**) [119] and (3) Oxford building dataset plus one million Flickr images (**Oxford5K+Flickr1M**). Table 5.1 shows the details of the three datasets.

Tattoo image dataset (tattoo) Tattoos have been commonly used in forensics and law enforcement agencies to assist in human identification. The tattoo image database used in our study consist of $101,745$ images, among which $61,745$ are tattoo images and the remaining

Data set	# images	# features	Size of descriptors
tattoo	101,745	10,843,145	3.4GB
Oxford5K	5,062	14,972,956	4.7GB
Oxford5K+Flickr1M	1,002,805	823,297,045	252.7GB

Table 5.1. Statistics of the datasets

$40,000$ images are randomly selected from the ESP dataset[1]. The purpose of adding images from the ESP dataset is to verify the capacity of the algorithms in distinguishing tattoo images from the other images. On average, about 100 Harris-Laplacian interesting points are detected for each image, and each key point is described by a 128-dimensional SIFT descriptor.

Oxford building dataset (Oxford5K) The Oxford building dataset consists of $5,062$ images. Although it is a small data set, we use it for evaluating the proposed algorithm for image retrieval mainly because it is one of the widely used benchmark datasets. When detecting keypoints for each image, we use both Harris-Laplacian and Hessian-Affine interesting point detectors and each key point is described by a 128-dimensional SIFT descriptor. Since the algorithms perform similarly with keypoints detected by the two methods, we only report the results based on the Harris-Laplacian detector. On average, about $3,000$ keypoints are detected for each image.

Oxford building dataset plus one million Flickr images (Oxford5K+Flickr1M) In this dataset, we first crawled Flickr.com to find about one million images of medium resolution and then added them into the Oxford building dataset. The same procedure is applied to extract keypoints from the crawled Flickr images.

[1] `http://www.gwap.com/gwap/gamesPreview/espgame/`

5.5.2 Implementation and Baselines

For the implementation of the proposed method, the kernel function (5.5) is used. The centers for the kernel are randomly selected from the datasets. We employ the FLANN library [2] to perform the efficient range search.

We select the bag-of-words model (**BoW**) [119] as our baseline which is a popular method for large scale image retrieval. It constructs the visual vocabulary by a hierarchical k-means algorithm that deploys FLANN library for efficient nearest neighbor search. We set the branching factor of hierarchical k-means to be 10 based on our experience. A forest of 8 randomized k-d trees is used in all experiments. We initialize cluster centers by randomly selecting a number of keypoints in the dataset. For the bag-of-words model, a state-of-the-art text retrieval method, Okapi BM25 [129] is used to compute the similarity between a query image and gallery images based on their bag-of-words representations. The inverted indices for both Okapi BM25 and the proposed retrieval model are stored in memory to make the retrieval procedure efficient.

5.5.3 Evaluation

In order to exam the efficiency of the proposed method, we measure the time spent on preprocessing as well as retrieval stage of the retrieval systems. For the proposed method, the preprocessing stage consists of three steps, i.e., randomly selecting a number of centers, identifying keypionts within the predefined range of the selected centers and computing weights $\boldsymbol{\alpha}$ of every image; For the baseline method, it consists of two steps, constructing visual vocabulary by clustering and mapping keypoints to visual words. For retrieval time, we report the averaged retrieving time of one query for the two methods. We emphasize that besides the retrieval time, the preprocessing time is also very important for an image retrieval system when it comes to a large collection of millions of images and the image collection is updated frequently. Take Flickr.com as an example, which is one of the most

[2] `http://www.cs.ubc.ca/~mariusm/index.php/FLANN/FLANN`

popular online photo sharing web sites, there are about 900,000 new images uploaded every day [98]. These images must be preprocessed in time in order to be used for retrieval, which requires the preprocessing of an algorithm be very efficient.

To evaluate the retrieval accuracy of the proposed method, we use two different metrics for the datasets. For tattoo image dataset, the retrieval accuracy is evaluated based on whether a system could retrieve images that share the tattoo symbol as in the query image. We adapt the evaluation metric termed Cumulative Matching Characteristics (CMC) score [105]in this study. For a given rank position k, its CMC score is computed as the percentage of queries whose matched images are found in the first k retrieved images. The CMC score is similar to recall, a common metric used in Information Retrieval. We use CMC score on the tattoo database because it is the most widely used evaluation metric in face recognition and forensic analysis.

For the Oxford building dataset and the Oxford building plus Flickr dataset, we follow [119] and evaluate the retrieval performance by Average Precision (AP) which is computed as the area under the precision-recall curve. In particular, an average precision score is computed for each of the 5 queries from a landmark specified in the Oxford building dataset, and these results are averaged to obtain the mean Average Precision (mAP) for each landmark.

5.5.4 Results on The Tattoo Image Dataset

We select 995 images as queries, and manually identify the gallery images that have the same tattoo symbols as the query images. We randomly select 100 images among the 995 query images and use them to train the optimal values for both λ and ρ. The learned parameter λ and ρ are used for the consequential experiments. The remaining images are used for testing.

We first show the retrieval results of both the proposed method and the bag-of-words method with the parameters tuned to achieve the best performance, and then show the sensitive of the proposed algorithm to the choice of parameters. Figure 5.1 give the retrieval performance of the two methods in CMC curves for the first 100 retrieved images. It is

	Preprocessing Time	Retrieval Time
Proposed	1.0h	0.02s
BoW	8.8h	0.009s

Table 5.2. The preprocessing and retrieval time of the two methods on tattoo dataset.

clear that the proposed algorithm significantly outperforms the bag-of-words model based approach.

The efficiency of the two methods are listed in the Table 5.2. For the preprocessing time, the proposed methods is about 8 times more efficient that the baseline methods; For the retrieval time, the proposed method is significantly slower than that of the bag-of-words model method. After carefully checking the implementation, we found that the difference in retrieval time is because the logarithm function used by the proposed method (5.17) takes a significantly longer time to compute than the simple addition and multiplication used by the BM25 model. In a real retrieval system, however, this disadvantage can be overcome by some engineering tricks. For example, a logarithm look up table can be built in advance and computing the logarithm of a value can be simplified as checking the lookup table. In fact, this trick is commonly used in the implementation of an automatic speech recognition system.

5.5.5 Parameter λ

Figure 5.2 shows the CMC curves of the proposed method with λ varied from 0.01 $\bar{n}$ to 100 $\bar{n}$, where $\bar{n}$ is the average number of keypoints in an image. In this experiment, we set the number of random centers to be one million, and ρ to be 0.6 $\bar{d}$, where $\bar{d}$ is the average distance between any two keypoints which is estimated from 1,000 randomly sampled keypoints from the collection. This result shows the performance of the proposed method is overall not sensitive to the choice of λ.

5.5.6 Parameter ρ

Figure 5.3 shows the CMC curves of the proposed method with ρ varied from 0.3 $\bar{d}$ to 1.1 $\bar{d}$. In this experiment, we again fixed the number of centers to be one million. From the figure we observe that with the exception of the smallest radius ρ (i.e., r = $0.3\bar{d}$), the retrieval system achieves similar performance for different values of ρ. This indicates that the proposed algorithm is in general insensitive to the choice of ρ as long as ρ is large enough compared to the average inter-points distance between keypoints. This result can be understood by the fact that in a high dimensional space, most data points are far from each other and as a result, unless we dramatically change the radius ρ, we do not expect the points within a distance ρ of the centers to change significantly.

5.5.7 Number of Random Centers

Figure 5.4 shows the performance of the proposed method with different number of randomly selected centers. The λ and ρ are selected to maximize the performance for the given number of centers. We clearly observe a significant increase in the retrieval accuracy when the number of centers is increased from 10K to 1M. This is not surprising because a large number of random centers usually results in a better discrimination between different SIFT keypoints and consequently leads to an improvement in the detection of similar images. A similar observation is also found when we run our retrieval system using the bag-of-words model approach which is consistent with the observation in [119].

5.5.8 Results on Oxford Building and Oxford Building + Flickr Datasets

Based to the observation from the experiments of tattoo image retrieval and the similar observation in [119], we use one million cluster/random centers in this experiment. The parameters of the proposed methods are set as the following based on our experiments done

	Proposed	BoW
Oxford5K	0.561	0.521
Oxford5K+Flickr1M	0.472	0.330

Table 5.3. mAP results of the proposed method and BoW method for Oxford5K building data set and Oxford5K+Flickr1M data set.

with tattoo images. We set $\rho = 0.6\bar{d}$, where $\bar{d}$ is the average inter-points distance that was estimated based on 1000 randomly sampled pairs. We set the parameter $\lambda = 10\bar{n}$ where $\bar{n}$ is the average number of key point in an image.

The mAP results of the proposed method and BoW method are listed in Table 5.3. In Figure 5.5, we show two examples of the queries and the retrieved images. Note that for the Oxford5K+Flickr1M dataset, we follow the experimental protocol in [119] by only using the cluster/random centers that are obtained from the images in the Oxford5K dataset. The results clearly show that the proposed method outperforms the bag-of-words model.

As expected the performance of the proposed method drops slightly when the 1M Flickr images are added to the Oxford5K dataset. In contrast, the bag-of-words based method suffers from a significant loss in the performance when we include one million images into the Oxford5K data set. We believe this difference in the performance is due to the fact that the visual content of the 1 million Flickr images is significantly different from that of the Oxford 5K images, i.e., the keypoints extracted from the Flickr images are generally far away from those in the Oxford5K images. As discussed before, the proposed method is robust to the outlying keypoints which makes it less sensitive to the inclusion of the Flickr1M dataset than the bag-of-words model. To verify this, we measure the distance between keypoints and centers for both Oxford5k data set and Oxford5k+Flickr1M data set. We find that for the Oxford building images, there are $\sim 8\%$ keypoints that are separated from any of the centers by a distance larger than $\rho = 0.6\bar{d}$. This percentage is increased to $\sim 24\%$ for the Flickr images, indicating that a large portion of keypoints from the Flickr images are significantly different from the keypoints from the Oxford building images.

	Proposed		BoW	
	preprocessing	retrieval	preprocessing	retrieval
Oxford5K	1.09h	0.8s	11.4h	0.08s
Oxford5K+Flickr1M	95h	1.3s	685h	0.9s

Table 5.4. Preprocessing and retrieval times of the two methods with one million cluster/random centers.

The preprocessing and retrieval times of the two algorithms are shown in Table 5.4. For preprocessing, we split the the Oxford5K+Flickr1M dataset into 82 subsets and each subset contains about 10,000,000 keypoints. These 82 subsets are processed separately on multiple machines, and are aggregated later to obtain the final result of key point quantization. The preprocessing time for 5K+1M dataset is estimated by the average processing time of each of the 82 subsets. It shows clearly that, for both the datasets, the proposed method is significantly more efficient, e.g. 10 times faster, in preprocessing time than the bag-of-words method. Combining the results of preprocessing time from the previous experiment on the tattoo image dataset, it is clear that the proposed method is significant more efficient than the BoW method in terms of preprocessing time, which makes it more applicable to large scale image retrieval.

For the retrieval time, we first compare the proposed algorithm to a naive retrieval method using 5.14 in which a linear scan of the whole database is required. On average, it takes more than 1100 seconds to retrieve a query for the naive retrieval method while the proposed efficient retrieval method only takes 0.8 second for one query. When comparing to the the BoW method, we observe that the proposed method is still significantly slower than the BoW model, due to the reason mentioned earlier.

5.6 Summary

In this chapter, we study the problem of dimensionality reduction with extending the data representation from a single vector to a set of unordered vectors. By describing each datum with a function which is estimated from its associated vectors, the task of dimensionality reduction becomes how to make an appropriate representation of each function to facilitate the computation of similarity between two data points and furthermore speedup the nearest neighbor search for a very large database. We study the problem in the context of large scale image retrieval with each image is represented by its local features. There are two main challenges in designing efficient search algorithms: one is how to efficiently estimate the density function for each image from its local features; the other is how to efficiently identify a subset of candidate images using the derived function representation of images. We have developed a statistical modeling approach and an efficient learning and retrieval algorithm for the two challenges. Our empirical results on three large-scale image retrieval tasks show that, comparing to the popular bag-of-words method, the proposed method is significantly more effective and more efficient in indexing images. The only disadvantage of the proposed method is it takes a longer time to retrieve images than the BoW model due to the high cost in computing the logarithm function.

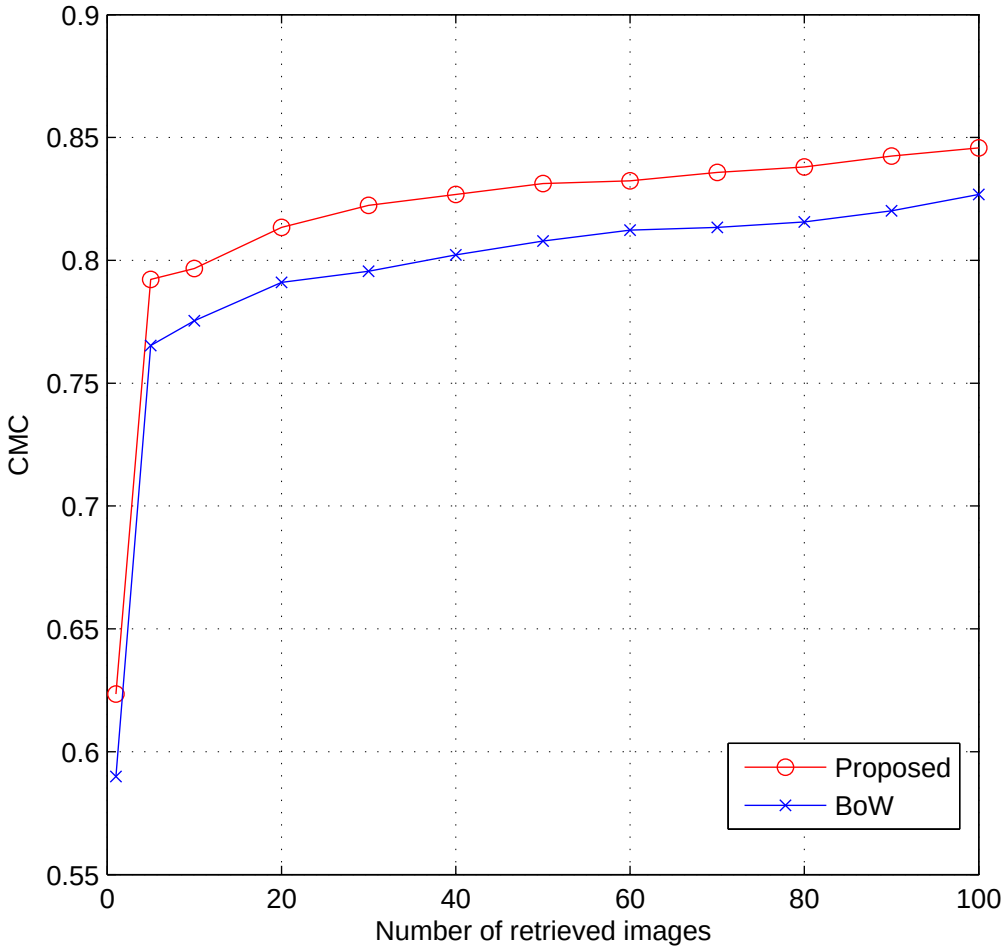


Figure 5.1. The CMC scores for tattoo image retrieval with one million cluster/random centers. For interpretation of the references to color in this and all other figures, the reader is referred to the electronic version of this dissertation.

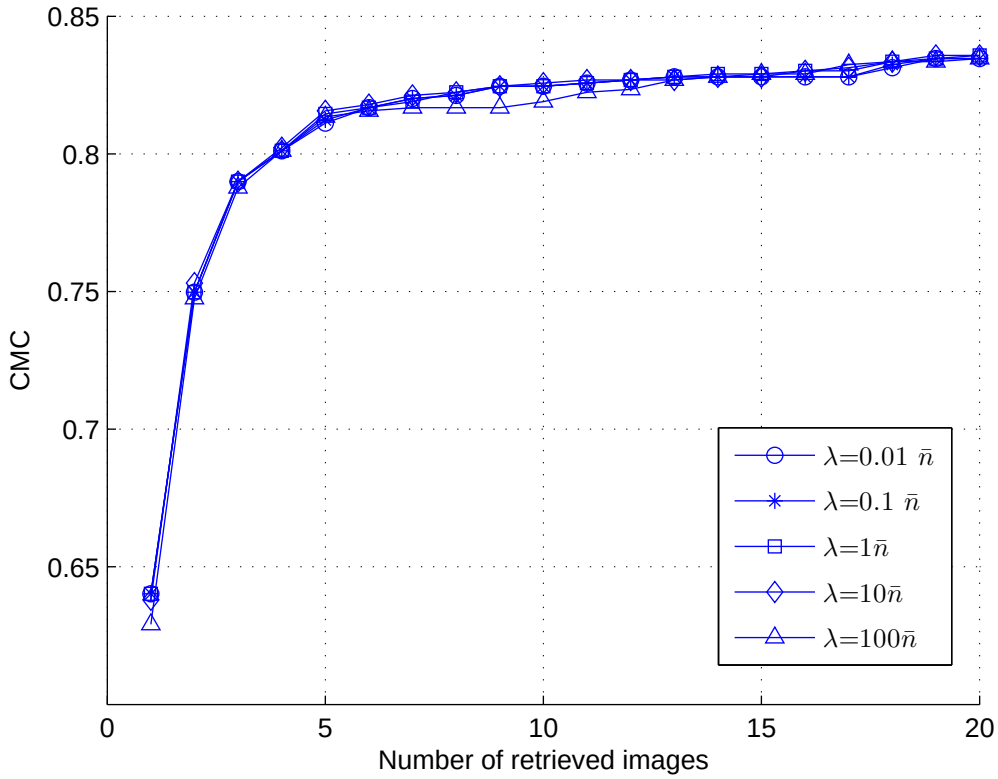


Figure 5.2. Results of the proposed method for tattoo image retrieval with different value of λ base on 1 million random centers with $\rho = 0.6\bar{d}$

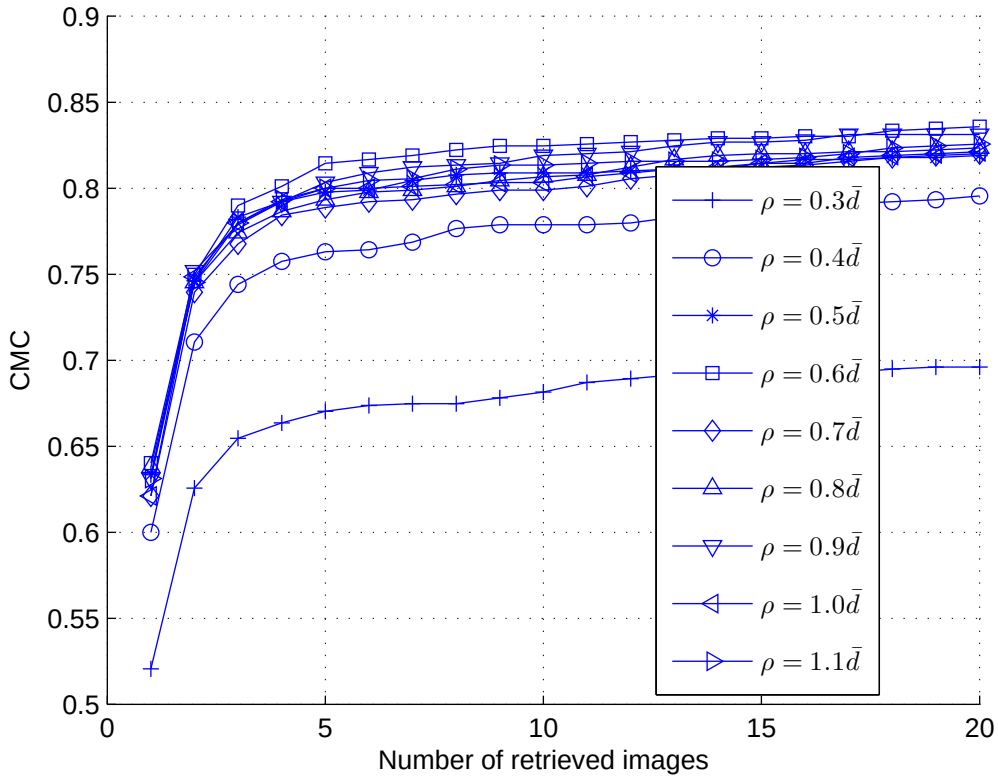


Figure 5.3. Results of the proposed method for tattoo image retrieval with different value of ρ base on 1 million random centers

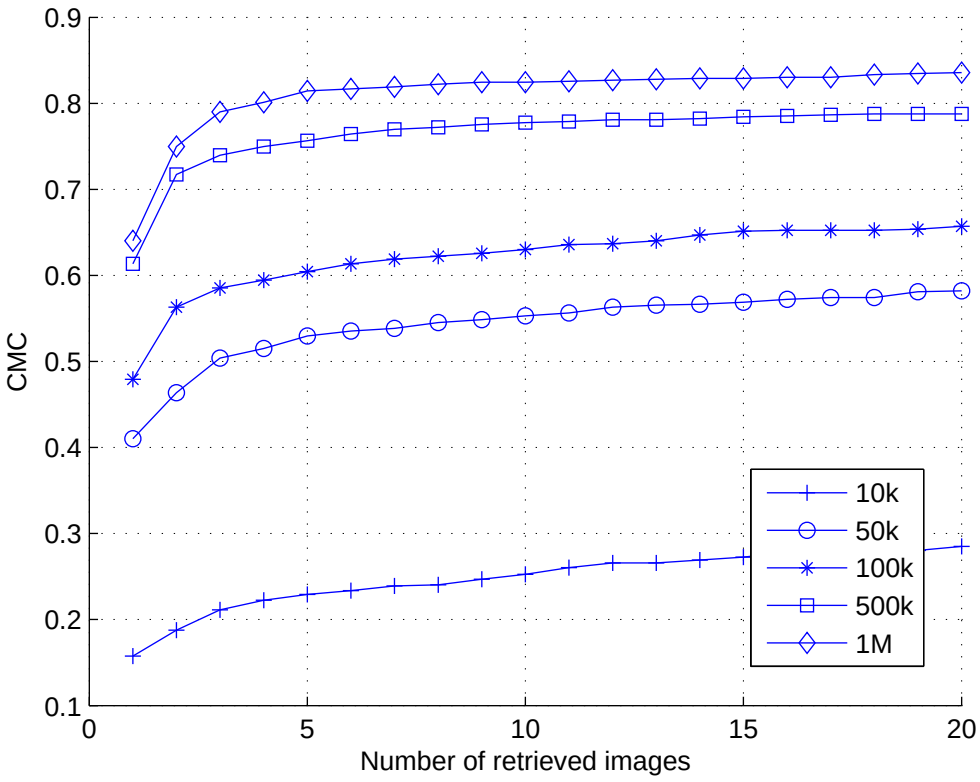


Figure 5.4. Results of the proposed method for tattoo image retrieval with different number of centers.

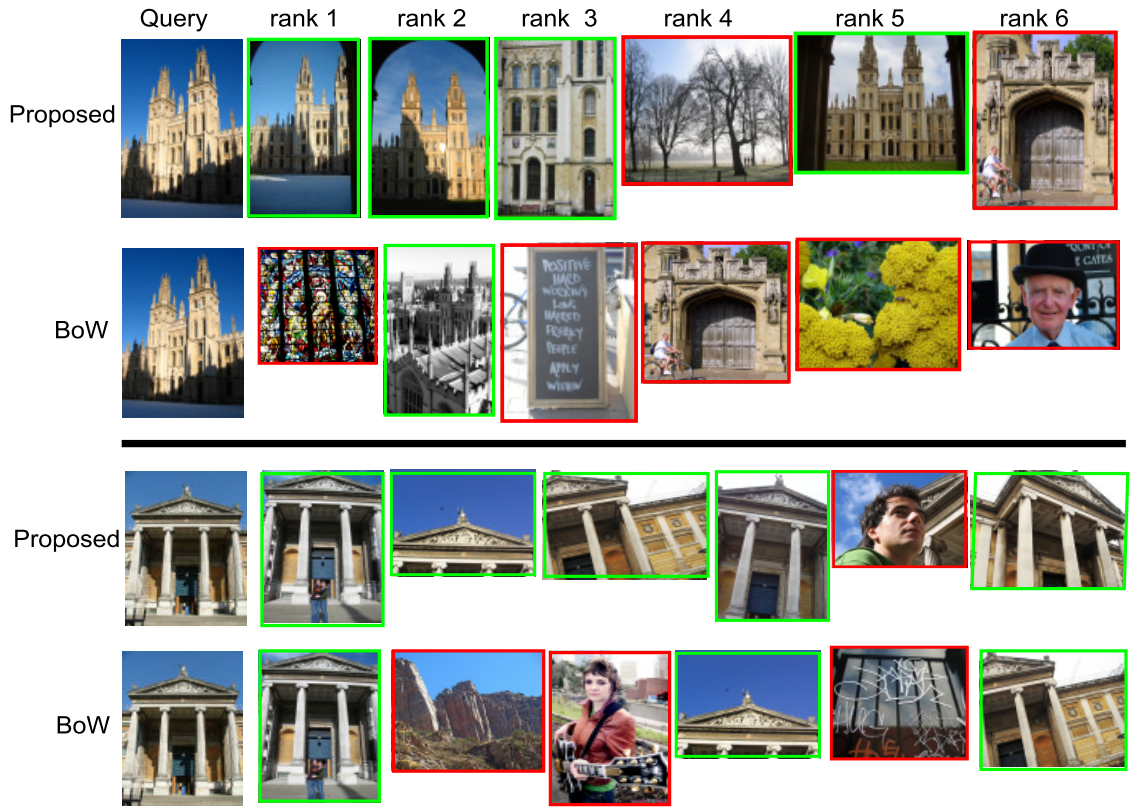


Figure 5.5. Examples of two queries (the first column) and the first six retrieved images. The first two rows give the retrieved results for the Oxford5K building database, and the next two rows give the retrieved results for Oxford5K+Flickr1M database. The correctly retrieved results are outlined in green and while irrelevant images are marked by the red color.

CHAPTER 6

Conclusion

People have been working on the problem of dimensionality reduction for more than a century and numerous algorithms have been proposed and studied. Recent research on dimensionality reduction has been fueled by the fast-paced development of statistical analysis, machine learning, as well as by the advances in convex optimization and the constant growth in sheer processing power. This dissertation continues along this line of research, and offers several novel contributions to the field:

- **Mixed label propagation for semi-supervised learning** Unlike the existing graph-based approaches that are only applicable to the positive similarity of examples, our framework is able to explore both positive and negative similarities simultaneously. The challenge of incorporating negative similarities comes from the fact that the negative similarity is non-transitive, and therefore cannot be propagated directly. In addition, objective functions employed by most graph-based approaches could be unbounded below when the similarity is negative, and therefore no optimal solutions can be found by minimizing the objective function. The key idea behind the proposed framework is to explore LDA, which is to minimize the inconsistency between the class assignments and the positive similarity of examples, and maximize the consistency between the class assignments and the negative similarity of examples. We present an efficient learning algorithm for the mixed label propagation that is based on the alternating optimization strategy and semi-definitive programming. Our empirical study with collaborative filtering shows that the proposed algorithm is effective in exploring negative similarities and outperforms both the label propagation approach and

state-of-the-art approaches for collaborative filtering

- **An semi-definite formulation for the second-order PCA-style algorithms** Most existing algorithms for second-order PCA-style algorithms require solving non-convex optimization problems, often leading to suboptimal performance. The key challenge in solving the optimization problem arises from the fact that they require identifying both the left and right projection matrices simultaneously that are dependent on each other. In our method, we try to avoid this difficulty by first factorizing each matrix into left and right parts, then extracting the left and right projection matrices from the sum of the left and right parts of all matrices. We begin with factorizing a single matrix which leads to the trace norm of a matrix. We then generalize it to the case of multiple matrices by extracting the largest common singular vectors of multiple matrices, and finally present a semi-definite programming formulation. The relationship between the proposed SDP formulation and that of GLRAM is also studied. Finally, we present an approximate algorithm to efficiently solve the related SDP problem and evaluate the proposed algorithm by image classification. Our experimental study shows that the performance of the proposed algorithm is better than or comparable to the state-of-the-art algorithms for SOPCA.

- **Dimensionality reduction for functional data representation and its application to large-scale image retrieval by local features** Since the number of local features used to represent each image is varied from one image to another, it is in general difficult to represent all the images in a vector space. One way to deal with the bag-of-features representation is to associate with each image a function which is estimated from its local features. The challenge of using functions to represent data is its high computational cost in estimating the similarity between functions. In this dissertation, we study the problem of conducting dimensionality reduction for the function representation of data. More specifically, we would like to find the appropriate

representation of the functions that could leads to efficient search of similar images for very large image databases. We have developed a statistical modeling approach and an efficient retrieval algorithm for the challenges which can efficiently estimate the functions of each image with its local features and further utilize the derived functions to efficiently identify a subset of candidate images which are visually similar to a given query. Our empirical studies on large-scale image retrieval tasks show that, the proposed methods is robust to the choice of the parameters and is significantly more effective and more efficient in indexing images comparing to the popular bag-of-words method.

Finally, besides the success of the studies conducted in this dissertation, there are a number of challenging problems that deserve further investigation in the future. For example, both of the mixed label propagation and the proposed convex formulation of SOPCA need to solve SDP problems which are computationally expensive. We may utilize some special structures of the involved SDP problems so that problem-specific SDP solvers might be built to solve the prolbems more efficiently. For the functional data dimensionality reduction, besides the simple basis function used in the proposed methods, we may be able to make a more intelligent choice of the basis functions so that the overall performance can be further improved.

APPENDICES

APPENDIX A

Notation

Some specific sets

$\mathbb{R}$	Real numbers
$\mathbb{R}^n$	Real n-vectors
$\mathbb{R}^{m\times n}$	Real $m \times n$ matrices
$\mathbb{R}_+, \mathbb{R}_{++}$	Nonnegative, positive real numbers
$\mathbb{R}^n_+, \mathbb{R}^n_{++}$	Nonnegative, positive real n-vectors
$\mathbf{S}^n$	Symmetric $n \times n$ matrices
$\mathbf{S}^n_+, \mathbf{S}^n_{++}$	Symmetric positive semi-definite, positive definite, $n \times n$ matrices

Vectors and matrices

$\mathbf{1}$	Vector with all components one
I	Identity matrix
$X^\top$	Transpose of matrix X
$\mathrm{tr}(X)$	Trace of matrix X
$X^\dagger$	Moore-Penrose or pesudo-inverse of matrix X
$\mathrm{rank}(X)$	Rank of matrix X

Norms and distances

$\|\cdot\|$	A norm
$\|\mathbf{x}\|_2$	Euclidean (of l_2-) norm of vector $\mathbf{x}$
$\|X\|_F$	Frobenius norm of matrix X
$\|X\|_{tr}$	Trace norm of matrix X

Generalized inequalities

$\mathbf{x} \preceq \mathbf{y}$	Componentwise inequality between vectors $\mathbf{x}$ and $\mathbf{y}$
$\mathbf{x} \prec \mathbf{y}$	Strick componentwise inequality between vectors $\mathbf{x}$ and $\mathbf{y}$
$X \preceq Y$	Matrix inequality between matrices X and Y
$X \prec Y$	Strict matrix inequality between matrices X and Y

APPENDIX B

Schur Complement

Consider a matrix $X \in S^n$ partitioned as:

$$X = \begin{pmatrix} A & B \\ B^\top & C \end{pmatrix}$$

where $A \in S^k$. If $detA \neq 0$, then matrix

$$S = C - B^\top A^{-1} B$$

is called the *Schur complement* of A in X.

APPENDIX C

Semi-Definite Programming

Semi-definite programming is a linear programming over the cone of semi-definite matrices. In comparison to standard linear programming, the variable of vector $\mathbf{x} \in \mathbb{R}^n_+$ is replaced by a variable of matrix $X \in S^n_+$.

Given m matrices $A_1, \ldots, A_m$, where each matrix $A_i \in S^n$, we define:

$$AX = \begin{pmatrix} \mathrm{tr}(A_1 X) \\ \vdots \\ \mathrm{tr}(A_m X) \end{pmatrix}$$

With this notation, the standard formulation of a semi-definite programming problem is defined as:

$$\begin{aligned} \min_{X \in S^n_+} \quad & \mathrm{tr}(CX) \\ \text{s. t.} \quad & AX = \mathbf{b} \\ & X \succeq 0 \end{aligned} \tag{C.1}$$

Similarly, if we define:

$$A^\top \mathbf{y} = \sum_{i=1}^{m} y_i A_i$$

the corresponding dual problem of the semi-definite programming problem in (C.1) is given as:

$$\begin{aligned} \max_{\substack{\mathbf{y} \in \mathbb{R}^m \\ Z \in \mathbf{S}^n_+}} \quad & \mathbf{b}^\top \mathbf{y} \\ \text{s. t.} \quad & A^\top \mathbf{y} + Z = C \\ & Z \succeq 0 \end{aligned} \tag{C.2}$$

In (C.1) and (C.2), the inequality $X \succeq 0$ and $Z \succeq 0$ are often called *linear matrix inequality* (LMI).

APPENDIX D

Trace Norm And Ky Fan k-Norm

The trace norm of a matrix X is defined as the sum of all singular values of X. It can also be defined as an optimization problem over all possible factorizations of matrix X, as shown in the following definition:

Definition 1. *The trace norm $\|X\|_{tr}$ of a matrix is given by any of the three quantities:*

1. $\min_{U,V} \left\{ \|U\|_F \|V\|_F : X = UV^\top \right\}$
2. $\min_{U,V} \left\{ \frac{1}{2}(\|U\|_F^2 + \|V\|_F^2) : X = UV^\top \right\}$
3. *The sum of the singular values of X*

Furthermore, If $X = U\Sigma V^\top$ is the singular value decomposition of X, then the matrices $U\sqrt{\Sigma}$ and $V\sqrt{\Sigma}$ minimize the first two quantities.

A very important property of trace norm is that it is the convex envelope of the function $\phi(X) = \text{rank}(X)$, on $C = \{X \in \mathbb{R}^{m\times n} | \|X\| \leq 1\}$ [42, 136]. Because of this property, trace norm is often used as the convex surrogate for the rank function in rank minimization [42]. Minimizing the trace norm can be converted to an SDP problem by the following lemma [42]:

Lemma 2. *For $X \in \mathbb{R}^{p\times q}$ and $t \in \mathbb{R}$, we have $\|X\|_{tr} \leq t$ if and only if there exist matrices $P \in \mathbb{R}^{p\times p}$ and $Q \in \mathbb{R}^{q\times q}$ such that:*

$$\begin{pmatrix} P & X \\ X^\top & Q \end{pmatrix} \succeq 0, \; tr(P) + tr(Q) \leq 2t$$

Using the above lemma, minimizing the trace norm of X is equivalent to the following problem:

$$\begin{aligned} \min_{\substack{P\in\mathbb{R}^{p\times p}\\ Q\in\mathbb{R}^{q\times q}}} \quad & \mathrm{tr}(P)+\mathrm{tr}(Q) \\ \text{s. t.} \quad & \begin{pmatrix} P & X \\ X^\top & Q \end{pmatrix} \succeq 0 \end{aligned} \tag{D.1}$$

It is straightforward to show that an optimal solution to (D.1) is $P = U\Sigma U^\top$ and $Q = V\Sigma V^\top$, where U and V are derived from the singular value decomposition of X, i.e., $X = U\Sigma V^\top$.

The Ky Fan k-norm $S_k(X)$ generalizes the concept of trace norm by computing the sum of the k largest singular values of matrix X. When X is a symmetric positive semi-definite matrix, $S_k(X)$ becomes the sum of the k largest eigenvalues of X.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Y. Altun, D. A. McAllester, and M. Belkin. Margin semi-supervised learning for structured variables. In *NIPS*, 2005.

[2] Arindam Banerjee, Srujana Merugu, Inderjit Dhillon, and Joydeep Ghosh. Clustering with bregman divergences. *Journal of Machine Learning Research*, (6):1705–1749, 2005.

[3] P. N. Belhumeour, J. P. Hespanha, and D. J. Kriegman. eigenfaces vs.fisherfaces: Recognition using class specific linear projection. *IEEE Trans. Pattern Anal. Mach. Intell*, 19(7):711–720, 1997.

[4] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Neural Information Processing Systems 14 (NIPS'2001)*, 2001.

[5] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):137–1396, 2003.

[6] M. Belkin and P. Niyogi. Semi-supervised learning on manifolds. *Machine Learning*, 56:209–239, 2004.

[7] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: a geometric framework for learning from examples. Technical Report TR-2004-06, University of Chicago Computer Science, 2006.

[8] R. Bellman. *Adaptive Control Process: A Guided Tour*. Princeton University Press, Princeton, 1961.

[9] Y. Bengio, J-F. Paiement, and P. Vincent. Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. In *NIPS*, 2004.

[10] M. Bernstein, V. de Silva, J. Langford, , and J. Tenenbaum. Graph approximations to geodesics on embedded manifolds. Technical report, Department of Psychology, Stanford University, 2000.

[11] M. Berry, M. Browne, A. Langville, and etc al. Algorithms and applications for approximate nonnegative matrix factorization. *Submitted to Computational Statistics and Data Analysis*, 2006.

[12] S. Yu J. Bi and J. Ye. Probabilistic interpretations and extensions for a family of 2d pca-style algorithms. In *KDD'2008 Workshop on Data Mining using Matrices and Tensors*, 2008.

[13] Christopher M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*, 61:611–622, 1999.

[14] J. Blitzer, K. Q. Weinberger, L. K. Saul, and F. C. N. Pereira. Hierarchical distributed representations for statistical language modeling. In *NIPS*, 2006.

[15] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[16] Matthew Brand and Matthew Brand. Incremental singular value decomposition of uncertain data with missing values. In *ECCV*, pages 707–720, 2002.

[17] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. Technical Report MSR-TR-98-12, Microsoft Research, 1998.

[18] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[19] Christopher J.C. Burges. Geometric methods for feature extraction and dimensional reduction: A guided tour. Technical Report MSR-TR-2004-55, Microsoft Research, 2004.

[20] Deng Cai, Xiaofei He, and Jiawei Han. Subspace learning based on tensor analysis. Technical Report UIUCCDCS-R-2005-2572, Department of Computer Science, UIUC, 2005.

[21] Miguel Á. Carreira-Perpiñá. A review of dimensional reduction techniques. Technical Report CS-96-09, Department of Computer Science, University of Sheffield, 1997.

[22] Chad Carson, Serge Belongie, Hayit Greenspan, and Jitendra Malik. Region-based image querying. In *Proceedings of IEEE Workshop on Content-Based Access of Image and Video Libraries*, pages 42–49, 1997.

[23] L. F. Chen, H. Y. M. Liao, J. C. Lin, M. D. Kao, and G. J. Yu. A new lda-based face recognition system which can solve the small sample size problem. *Pattern Recognit.*, 33(10):1713–1726, 2000.

[24] M. Chu, F. Diele, and S. Plemmons, R.and Ragni. Optimality, computation, and interpretations of nonnegative matrix factorization, 2004.

[25] W. Chu and Z. Ghahramani. Preference learning with gaussian processes. In *ICML'05*, 2005.

[26] A. Cichocki, S. Amari, K. Siwek, T. Tanaka, and et al. Icalab toolboxes. `http://www.bsp.brain.riken.jp/ICALAB`.

[27] A. Cichocki, R. Zdunek, and S. Amari. Csiszars divergences for non-negative matrix factorization: Family of new algorithms. In *Proc. 6th International Conference on Independent Component Analysis and Blind Signal Separation*, 2006.

[28] T. Cox and M. Cox. *Multidimensional Scaling*. Chapman And Hall, London, 1994.

[29] G. Csurka, C. Bray, C. Dance, and L. Fan. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision, ECCV*, 2004.

[30] M. Cuturi and J-P Vert. Semigroup kernels on finite sets. In *Advances in Neural Information Processing Systems*, 2005.

[31] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, 2004.

[32] D. de Ridder, M. Loog, and M.J.T. Reinders. Local fisher embedding. In *17th International Conference on Pattern Recognition (ICPR2004)*, 2004.

[33] V. de Silva and J. B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In *Neural Information Processing Systems (NIPS)*, pages 705–712, 2002.

[34] I. Dhillon and S. Sra. Generalized nonnegative matrix approximations with bregman divergences. In *Proceeding of the Neural Information Processing Systems (NIPS)*, 2005.

[35] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1:269–271, 1959.

[36] Chris Ding and Xiaofeng He. K-means clustering via principal component analysis. In *ICML '04: Proceedings of the 23rd international conference on Machine learning*, 2004.

[37] Chris Ding, Tao Li, and Michael I. Jordan. Convex and semi-nonnegative matrix factorizations. Technical Report 60428, Lawrence Berkeley National Laboratory, 2006.

[38] Chris Ding and Jieping Ye. Two-dimensional singular value decomposition (2dsvd) for 2d maps and images. In *SDM*, 2005.

[39] D. L. Donoho and C. Grimes. When does isomap recover natural parameterization of families of articulated images? Technical Report 2002-27, Department of Statistics, Stanford University, 2002.

[40] John Eakins and Margaret Graham. University of northumbria at newcastle. Technical Report 39, IBM Research, 1999.

[41] Theodoros Evgeniou, Massimiliano Pontil, and Tomaso Poggio. Regularization networks and support vector machines. *Advanced In Computational Mathematics*, 13:1C50, 2000.

[42] M. Fazel, H. Hindi, and S. Boyd. A rank minimization heuristic with application to minimum order system approximation. In *Proceedings American Control Conference*, 2001.

[43] L. Finesso and P. Spreij. Approximate nonnegative matrix factorization via alternating minimization. In *Proc. 16th International Symposium on Mathematical Theory of Networks and Systems*, 2004.

[44] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Ann. Eugenics*, 7:179–188, 1936.

[45] Imola K. Fodor. A surveyof dimensional reduction techniques, 2002.

[46] Y. Freund, R. D. Lyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. In *ICML'98*, 1998.

[47] Zoubin Ghahramani and Geoffrey E. Hinton. The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, 21 1996.

[48] Andrew Goldberg, Xiaojin Zhu, and Stephen Wright. Dissimilarity in graph-based semi-supervised classification. In *AISTATS*, 2007.

[49] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1996.

[50] E. Gonzalez and Y. Zhang. Accelerating the lee-seung algorithm for nonnegative matrix factorization. Technical Report TR-05-02, Rice University, 2005.

[51] R. L. Gorsuch. *Factor analysis (2nd. ed.)*. Hillsdale, N.J.: Erlbaum, 1983.

[52] Kristen Grauman and Trevor Darrell. The pyramid match kernel: Efficient learning with sets of features. *J. Mach. Learn. Res.*, 8:725–760, 2007.

[53] Ming Gu, Stanley, C. Eisenstat, and In O. A stable and fast algorithm for updating the singular value decomposition. Technical report, Department of Computer Science, Yale University, 1994.

[54] D. Guillamet, M. Bressan, and J. Vitria. A weighted non-negative matrix factorization for local representations. In *Proc. 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition.*, 2001.

[55] Y. Guo, T. Hastie, and R. Tibshirani. Regularized linear discriminant analysis and its application in microarrays. *Biostatistics*, 8(1):86–100, 2007.

[56] J. Ham, D. D. Lee, S. Mika, and B. Scholköpf. A kernel view of the dimensionality reduction of manifolds. Technical Report TR-110, Max Planck Institute for Biological Cybernetics, 2003.

[57] T. J. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84:502–516, 1989.

[58] X. He and P. Niyogi. Locality preserving projections. In *Neural Information Processing Systems 16 (NIPS'2003)*, 2003.

[59] M. Herbster, M. Pontil, and L. Wainer. Online learning over graphs. In *ICML '05*, 2005.

[60] K Hirata and T Kato. Query by visual example - content-based image retrieval. In *Third International Conference on Extending Database Technology*, 1992.

[61] T. Hofmann. Gaussian latent semantic models for collaborative filtering. In *SIGIR'03*, 2003.

[62] T. Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):89–115, 2004.

[63] P. Hoyer. Non-negative matrix factorization with sparseness constraints. *J. of Machine Learning Research*, 5:1457–1469, 2004.

[64] J. J. Hull. A database for handwritten text recognition research. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(5):550–554, 1994.

[65] Aapo Hyvärinen. Independent component analysis. *Neural Computing Surveys*, 2, 2001.

[66] K. Inoue and K. Urahama. Equivalence of non-iterative algorithms for simultaneous low rank approximations of matrices. In *CVPR '06*, pages 154–159, 2006.

[67] J.E. Jackson. *A Users Guide to Principal Components*. New York: John Wiley and Sons, 1991.

[68] A K Jain, J Mao, and K Mohiuddin. Artificial neural networks: A tutorial. *Computer*, (29), 1996.

[69] Shuiwang Ji and Jieping Ye. Generalized linear discriminant analysis: A unified framework and efficient model selection. *Neural Networks, IEEE Transactions on*, 19:1768–1782, 2008.

[70] T. Joachims. Transductive learning via spectral graph partitioning. In *ICML'03*, 2003.

[71] Richard A. Johnson and Dean W. Wichern. *Applied Multivariate Statistical Analysis(5th Edition)*. Pearson Education, 2001.

[72] I.T. Jolliffe. *Principal Component Analysis, Second Edition*. Springer-Verlag, 2002.

[73] M. C. Jones and R. Sibson. What is projection pursuit? *Journal of the Royal Statistical Society*, 19:1–18, 1987.

[74] Tuomas Kärnä. Functional data dimensionality reduction for machine learning. Master's thesis, Department of Electrical Communications Engineering, Helsinki University of Techonology, 2007.

[75] Yan Ke, Rahul Sukthankar, and Larry Huston. Efficient near-duplicate detection and sub-image retrieval. In *Proceeding of ACM Multimedia*, pages 869–876, 2004.

[76] J.J. Kivinen, E.B. Sudderth, and M.I. Jordan. Learning multiscale representations of natural scenes using dirichlet processes. In *ICCV*, 2007.

[77] Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*. to appear (accepted June 2008).

[78] R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. In *In Proceedings of the Twenty First International Conference on Machine Learning(ICML02)*, 2002.

[79] Risi Imre Kondor and Tony Jebara. A kernel between sets of vectors. In *ICML*, 2003.

[80] O Kouropteva, O Okun, A Hadid, M Soriano, S Marcos, and M Pietikäinen. Beyond locally linear embedding algorithm. Technical Report MVG-01-2002, University of Oulu, Machine Vision Group, Information Processing Laboratory, 2002.

[81] O Kouropteva, O Okun, and M Pietikäinen. Selection of the optimal parameter value for the locally linear embedding algorithm. In *Proc. of the 1st International Conference on Fuzzy Systems and Knowledge Discovery (FSKD02)*, 2002.

[82] B. Kulis, S. Basu, I. Dhillon, and R. J. Mooney. Semi-supervised graph clustering: A kernel approach. In *ICML'05*, 2005.

[83] John Lafferty and Guy Lebanon. Information diffusion kernels. In *Advances in Neural Information Processing Systems 15*, 2002.

[84] M. H. Law and A. K. Jain. Incremental nonlinear dimensionality reduction by manifold learning. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, 28(3):377–391, 2006.

[85] M. H. Law and A. K. Jain N. Zhang. Nonlinear manifold learning for data stream. In *Proceedings of SIAM Data Mining*, pages 33–44, 2004.

[86] S. Lazebnik and et al. A sparse texture representation using affine-invariant regions. In *CVPR*, 2003.

[87] D. Lee and H. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[88] D. Lee and H. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, 2001.

[89] Vincent Lepetit, Pascal Lagger, and Pascal Fua. Randomized trees for real-time keypoint recognition. In *CVPR*, 2005.

[90] C.-J. Lin. On the convergence of multiplicative update algorithms for non-negative matrix factorization. Technical report, Department of Computer Science, National Taiwan University., 2005.

[91] C.-J. Lin. Projected gradient methods for non-negative matrix factorization. Technical Report ISSTECH-95-013, Department of Computer Science, National Taiwan University., 2005.

[92] F Liu and R W Picard. Periodicity, directionality and randomness: Wold features for image modelling and retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1996.

[93] T. Liu, A. Moore, A. Gray, and K. Yang. An investigation of practical approximate nearest neighbor algorithms. In *NIPS*, 2004.

[94] D. Lowe. Distinctive image features form scale-invariant keypoints. In *International Journal of Computer Vision*, 2004.

[95] Haiping Lu, K.N Plataniotis, and A.N. Venetsanopoulos. Mpca: Multilinear principal component analysis of tensor objects. *Neural Networks, IEEE Transactions on*, 19:18–39, 2008.

[96] KAPLAN L. M., MURENZI R., and NAMUDURI K. R. Fast texture database retrieval using extended fractal features. In *Storage and Retrieval for Image and Video Databases VI*, pages 162–173, 1998.

[97] W Y Ma and Manjunath B S. A texture thesaurus for browsing large aerial photographs. *Journal of the American Society for Information Science*, 1998.

[98] Pavan K. Mallapragada, Rong Jin, and A. K. Jain. Online visual vocabulary pruning using pairwise constraints. In *CVPR*, 2010.

[99] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval,*. Cambridge University Press, 2008.

[100] Benjamin Marlin. Modeling user rating profiles for collaborative filtering. In *n Advances in Neural Information Processing Systems*, 2004.

[101] A. M. Martinez and M. Zhu. where are linear feature extraction methods applicable? *IEEE Trans. Pattern Anal. Mach. Intell*, 27(12):1934–1944, 2005.

[102] A.M. Martinez and R. Benavente. The ar face database. Technical Report 24, Computer Vision Center at the U.A.B, 1998.

[103] R Mehrotra and J E Gary. Similar-shape retrieval in shape data management. *IEEE Computer*, 1995.

[104] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. In *International Journal of Computer Vision*, 2004.

[105] H. Moon and P. J. Phillips. Computational and performance aspects of pca-based face recognition algorithms. *Perception*, 30:303–321, 2001.

[106] P. Moreno, P. Ho, and N. Vasconcelos. A kullback-leibler divergence based kernel for svm classification in multimedia applications. In *Neural Information Processing Systems*, 2003.

[107] Pedro J. Moreno and et al. A kullback-leibler divergence based kernel for svm classification in multimedia applications. In *NIPS*, 2003.

[108] M.Stricker and D.Ballard. Similarity of color images. 1995.

[109] M.Swain and D. Ballard. Color indexing. *Journal of Computer Vision*, 1991.

[110] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Applications*, 2009.

[111] John Von Neumann. Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100(1):295–320, 1928.

[112] Carlton W. Niblack, Ron Barber, Will Equitz, Myron D. Flickner, Eduardo H. Glasman, Dragutin Petkovic, Peter Yanker, Christos Faloutsos, and Gabriel Taubin. The qbic project: querying images by color, texture and shape. Technical Report RJ-9203, IBM Research, 1993.

[113] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006.

[114] P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5:111–126, 1994.

[115] E. Parzen. On estimation of a probability density function and mode. *Ann. Math. Stat*, 1962.

[116] V. Pauca, F. Shahnaz, and M. Berry. Text mining using non-negative matrix factorizations. In *Proceedings of the Fourth SIAM International Conference on Data Mining*, 2004.

[117] D. M. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles. Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach. In *Proc. UAI*, 2000.

[118] K. Pettis, T. Bailey, A. Jain, and R. Dubes. An intrinsic dimensionality estimator from near-neighbor information. *IEEE Transactions on Pattern Analysis and Machine Intelligence(PAMI)*, 1(1):25–37, 1979.

[119] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, 2007.

[120] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *CVPR*, 2008.

[121] M. Polito and P. Perona. Grouping and dimensionality reduction by locally linear embedding. In *Neural Information Processing Systems(NIPS)*, 2001.

[122] J M Ponte and W B Croft. A language modeling approach to information retrieval. *Research and Development in Information Retrieval*, 1998.

[123] J. O. Ramsay and B.W. Silverman. *Functional data analysis 2nd ed.* Springer, 2005.

[124] J. Rennie and N. Srebro. Fast maximun margin matrix factorization for collaborative prediction. In *ICML'05*, 2005.

[125] Jason Rennie and Nathan Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *22nd International Conference on Machine Learning (ICML)*, 2005.

[126] P. Resnick, N. Iacovou, M.Sushak, P. Bergstrom, and J.Riedl. Grouplense: An open architecture for collaborative filtering of netnews. In *Proc. Computer Supported Collaborative Work Conference*, 1994.

[127] D. Ridder and Duin R.P.W. Locally linear embedding for classification. Technical Report PH-2002-01, Pattern Recognition Group, Dept. of Imaging Science And Technology, Delft University of Technology, 2002.

[128] D Ritter, O Kouropteva, O Okun, M Pietikäinen, and RPW Duin. Supervised locally linear embedding. In *Artificial Neural Networks and Neural Information Processing(ICANN/ICONIP)*, 2003.

[129] Stephen E. Robertson, Steve Walker, and Micheline Hancock-Beaulieu. Okapi at trec-7. In *Proceedings of the Seventh Text REtrieval Conference*, 1998.

[130] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323C–2326, 2000.

[131] Sam Roweis. EM algorithms for PCA and SPCA. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. The MIT Press, 1998.

[132] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., 1986.

[133] F. Samaria and A. Harter. Parameterisation of a stochastic model for human face identification. In *2nd IEEE Workshop on Applications of Computer Vision*, 1994.

[134] L. K. Saul and S. T. Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, 4:119–155, 2003.

[135] L. K. Saul, K. Q. Weinberger, J. H. Ham, F. Sha, and D. D. Lee. Spectral methods for dimensionality reduction. In O. Chapelle B. Schoelkopf and A. Zien, editors, *Semisupervised Learning*. MIT Press, 2006.

[136] B. Schölkopf, A. Smola, and K. R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. Technical Report TR-44, Max Planck Institute for Biological Cybernetics, 1996.

[137] B. Schölkopf, A. Smola, and K. R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[138] Bernhard Scholköpf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[139] N. SEBE and LEW. Color based retrieval. *Pattern Recognition Letters*, 2001.

[140] F. Shahnaz, M. Berry, V. Pauca, and R. Plemmons. Document clustering using nonnegative matrix factorization. In *Information Processing And Management*, 2006.

[141] Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. MIT Press, 2006.

[142] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[143] C. Silpa-Anan and R. Hartley. Localization using an imagemap. In *Proceedings of the 2004 Australasian Conference on Robotics & Automation*, 2004.

[144] Chanop Silpa-Anan and Richard Hartley. Optimised kd-trees for fast image descriptor matching. In *CVPR*, 2008.

[145] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, 2003.

[146] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *International Conference on Computer Vision*, 2003.

[147] N. Srebro, J.D. Rennie, and T.S. Jaakkola. Maximun-margin matrix factorization. In *NIPS*, 2005.

[148] Nathan Srebro. *Learning with Matrix Factorization.* PhD thesis, Massachusetts Institute of Technology, 2004.

[149] Nathan Srebro, Jason D. M. Rennie, and Tommi S. Jaakola. Maximum-margin matrix factorization. In *Proceeding of the Neural Information Processing Systems (NIPS)*, 2005.

[150] J. Sun, S. Boyd, L. Xiao, and P. Diaconis. The fastest mixing markov process on a graph and a connection to a maximum variance unfolding problem. *SIAM Review, submitted*, 2006.

[151] Hideyuki Tamura, Shunji Mori, and Takashi Yamawaki. Textural features corresponding to visual perception. *EEE Transactions on Systems, Man and Cybernetics*, 1978.

[152] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[153] M. E. Topping and C. M. Bishop. Probablistic principal component analysis. Technical Report NCRG/97/010, Neural Computing Research Group, Dept of Computer Science And Applied Mathematics, Aston University, 1997.

[154] M. E. Topping and C. M. Bishop. Mixtures of probablistic principal component analyzers. *Neural Computation*, 11(2):443–482, 1999.

[155] M. E. Topping and C. M. Bishop. Probablistic principal component analysis. *Journal of the royal Statistical Society*, 61(3), 1999.

[156] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996.

[157] Y. Wang, Y. Jiar, C. Hu, and M. Turk. Fisher non-negative matrix factorization for learning local features. In *Asian Conference on Computer Vision.*, 2004.

[158] G A Watson. On matrix approximation problems with ky fan k norms. *Numerical Algorithms*, (5):5–263, 1993.

[159] K. Q. Weinberger, B. D. Packer, and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, Barbados, January 2005.

[160] K. Q. Weinberger and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-04)*, volume 2, pages 988–995, Washington D.C., 2004.

[161] K. Q. Weinberger and L. K. Saul. An introduction to nonlinear dimensionality reduction by maximum variance unfolding. In *AAAI*, Boston, August 2006.

[162] K. Q. Weinberger and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision*, 70(1):77–90, 2006.

[163] K. Q. Weinberger, F. Sha, and L. K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings of the Twenty First International Conference on Machine Learning (ICML-04)*, pages 839–846, Banff, Canada, 2004.

[164] Kilian Weinberger, Fei Sha, Qihui Zhu, and Lawrence Saul. Graph laplacian methods for large-scale semidefinite programming, with an application to sensor localization. In B. Schölkopf, J. Platt, and Thomas Hofmann, editors, *Advances in Neural Information Processing Systems 19*, Cambridge, MA, 2007. MIT Press.

[165] Christopher K. I. Williams. On a connection between kernel PCA and metric multidimensional scaling. In *NIPS*, pages 675–681, 2000.

[166] J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *ICCV*, 2005.

[167] Lin Xiao, Jun Sun, and Stephen Boyd. A duality view of spectral methods for dimensionality reduction. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 1041–1048, New York, NY, USA, 2006. ACM Press.

[168] J. Yang, D. Zhang, A.F. Frangi, and JY. Yang. Two-dimensional pca: a new approach to appearance-based face representation and recognition. *PAMI*, 26(1):131–137, 2004.

[169] Liu Yang. Distance metric learning: A comprehensive survey, 2007.

[170] Xin Yang, Haoying Fu, Hongyuan Zha, and Jesse Barlow. Semi-supervised nonlinear dimensionality reduction. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, 2006.

[171] J. Ye. Characterization of a family of algorithms for generalized discriminant analysis on undersampled problems. *J. Mach. Learn. Res.*, 6:483–502, 2005.

[172] J. Ye. Generalized low rank approximations of matrices. *Machine Learning*, 61:167–191, 2005.

[173] W. Zhao, R. Chellappa, and P. Phillips. Subspace linear discriminant analysis for face recognition. Technical Report CAR-TR-914, Cntr. Autom. Res., Univ. Maryland, 1999.

[174] D. Zhou, B. Schölkopf, and T. Hofmann. Semisupervised learning on directed graphs. In *NIPS*, 2005.

[175] D.Y. Zhou, O. Bousquet, T. Navin Lal, J. Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *NIPS*, 2004.

[176] X. Zhu, Z. Ghahramani, and J.D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML'03*, 2003.

[177] Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report TR 1530, Computer Sciences, University of Wisconsin Madison, 2008.