Fall 2014

Relation among images: Modelling, optimization and applications

Bin Shen
Purdue University

Graduate School Form 30
(Revised 08/14)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By SHEN, BIN

Entitled RELATION AMONG IMAGES: MODELLING, OPTIMIZATION AND APPLICATIONS

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

Jan P. Allebach

Mikhail J. Atallah

Greg N. Frederickson

Yi Wu

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification/Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Approved by Major Professor(s): Jan P. Allebach

Approved by: Sunil Prabhakar/William J. Gorman 12/04/2014

Head of the Department Graduate Program Date

RELATION AMONG IMAGES:

MODELLING, OPTIMIZATION AND APPLICATIONS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Bin Shen

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2014

Purdue University

West Lafayette, Indiana

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Shen, Bin Ph.D., Purdue University, December 2014. Relation among Images: Modelling, Optimization and Applications. Major Professors: Jan P. Allebach and Mikhail J. Atallah.

In the last two decades, the increasing popularity of information technology has led to a dramatic increase in the amount of visual data. Many applications are developed by processing, analyzing and understanding such increasing data; and modelling the relation among images is fundamental to success of many of them. Examples include image classification, content-based image retrieval and face recognition. Given signatures of images, there are many ways to depict the relation among them, such as pairwise distance, kernel function and factor analysis. However, existing methods are still insufficient as they suffer from many real factors such as misalignment of images and inefficiency from nonlinearity. This dissertation focuses on improving the relation modelling, its applications and related optimization. In particular, three aspects of relation modelling are addressed: 1. Integrate image alignment into the relation modelling methods, including image classification and factor analysis, to achieve stability in real applications. 2. Model relation when images are on multiple manifolds. 3. Develop nonlinear relation modelling methods, including tapering kernels for sparsification of kernel-based relation models and developing piecewise linear factor analysis to enjoy both the efficiency of linear models and the flexibility of nonlinear ones. We also discuss future directions of relation modelling in the last chapter from both application and methodology aspects.

1 INTRODUCTION

In the last two decades, the increasing popularity of smart phones and consumer cameras has led to a dramatic increase in the amount of visual data. Furthermore, the image and video sharing web sites, such as Facebook, Flickr and YouTube, make these data easily available online and penetrate our everyday lives. Processing and further analyzing such increasing data becomes a more and more important and critical research area. Researchers are attracted to this area hoping that computer programs would be able to interpret the data and mine knowledge from them automatically or with little human interaction.

There are many topics drawing researchers' attention, such as boundary detection, image segmentation, structure from motion, image clustering/classification, image retrieval, face recognition and intelligent video surveillance, which process, analyze, or understand the content of images/videos. For many, if not most, of these application, modelling the relation among images is a key and fundamental component.

1.1 Role of Image Relation

Here several example applications, where image relations play critical roles, are briefly reviewed.

Content-based image retrieval aims to return a set of visually similar images for a given query image, which requires the relation between the query image and images in database. An engineered metric which reflects visual similarity is desired to rank the images in database according to visual similarity between them and the query image. Also, in many real image retrieval systems (not limited to content-based ones), the returned images should not only have the visual similarity but also have some diversity. Thus, nearly duplicate images should be removed from the final list shown to end users. A similar task is image clustering, which clusters images according to some certain metric, ideally visual similarity or semantic meaning. The images that look or are semantically similar to each other should belong to a same cluster.

In face recognition, a computer program automatically identifies or verifies a query face image given a database of labeled human faces. How to model the relation between a query face and the images in the database is the core component of face recognition algorithms.

Object tracking aims to locate a given object in each of the frames in a video sequence. Many algorithms, varying from the simple feature point matching method to nonrigid object tracking, have been proposed to deal with this task by making a decision in each frame considering mainly the relation between consecutive frames or the current frame and the frame where the object model is constructed. For example, the mean-shift tracking algorithm treats tracking as a mode seeking problem, and the model of the target is constructed usually from the first frame, and then mean-shift method searches the optimal mode in each of following frames.

1.2 Image Relation Modelling

Assume we have signatures/features of images, there are many components to help depict the relation among them, such as various distances, kernel functions and factor analysis. Ideally, these techniques should be consistent with the intent of related applications, computationally efficient and robust to real-world noise. There are many other methods that can be used to measure the relation among images, such as feature point matching and region of interest comparison. Only a few fundamental and representative ones are explored in this chapter.

Pairwise distance is one of the most popular and intuitive approaches, including l_p distance, Hausdorff distance, K-L divergence and earth mover's distance. In this case, images are usually represented as vectors in the feature space, and the distances between pairs of them are calculated.

Different from traditional distance measure, which is usually computed in the original feature space, kernels [1–3], which measure the relation between images in implicit

Hilbert space, have been later applied to image processing tasks, such as image clustering/classification, face recognition and object tracking. They are incorporated into a number of machine learning algorithms [2–7], such as matrix factorization, k-means clustering, and support vector machine, to form new powerful methods, i.e., the so-called kernel methods. In kernel methods, original input features are implicitly projected to a high, possibly infinite, dimensional space via a kernel-induced potentially nonlinear mapping, such that more useful features hidden in the original data can be utilized.

Besides kernel trick, factorization techniques such as Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), Independent Component Analysis (ICA) and Nonnegative Matrix Factorization (NMF) are also able to convert images to another space, which is believed to be more suitable for further analysis than the original feature space, and thus have been widely used in many image processing and computer vision tasks. After factorization, images have a compact new representation, which saves storage space and computational cost for further processing, and the relation among images can be more easily or accurately discovered or evaluated. For example, the user will easily determine whether two images belong to the same cluster by checking whether they share a same factor. Another example is that the similarity/distance can be calculated in the new low-dimensional space. Among these techniques, NMF [8] is popularly adopted since image data are usually nonnegative in nature.

1.3 Challenges and Opportunities

Challenges and opportunities coexist in the area of image relation modelling. This dissertation focuses on the following three aspects.

1.3.1 Misalignment in images

Many relation modelling techniques such as pairwise distance approach and factor analysis are widely used in many image processing applications such as face recognition and image clustering. However, the performance of most, if not all, of these modelling meth-

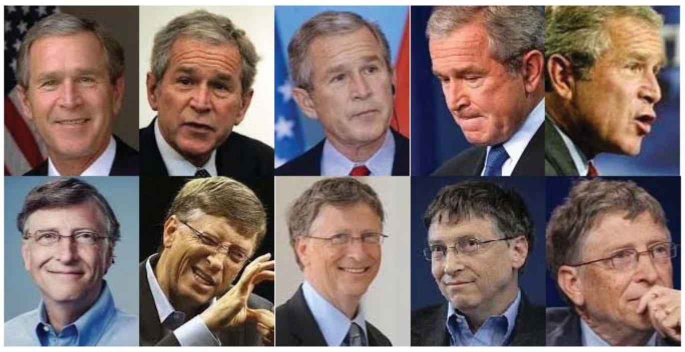


Figure 1.1.: Examples of misaligned celebrity face images.

ods depend heavily on the good alignment of the input samples, and they assume that the images are well aligned. Unfortunately, this assumption hardly holds in many real image processing applications [9], especially when the raw pixels are treated as features. Real image samples, such as face images, are usually with various poses and scales, and have misalignment due to many uncontrollable factors in the collecting process. Figure 1.1 shows some example face images of celebrities collected via Google image search, from which we can easily see that the face images are hardly aligned. Most largely available images on the Internet are misaligned as shown in the figure. Thus, alignment becomes a critical problem to be solved for the following processing such as factorization. Lack of reliable and efficient alignment algorithms for a large number of images makes it difficult for many image analysis tasks such as face recognition, image classification.

A lot of work has been done toward the image alignment problem, where all the images of an object or objects of interest are aligned to a fixed canonical template. The authors of [10] proposed a congealing algorithm to align images by minimizing the sum of entropies of pixel values at different pixel locations. A least squares congealing algorithm was proposed in [11] by minimizing the sum of the squared difference between all pairs. In [12], the proposed algorithm aligns the images via approximated rank minimization. In [9], the sparse representation is learned while implicitly aligning the images by inferring the coefficient of the first derivatives. [13] assumes input vectors that are translationally shifted versions of each other, and includes a shift-invariant representations into the N-

MF formulation. In [14] the authors make the algorithm scale and rotation invariant by transforming the images into log-polar coordinate system. In [15] a subspace is learned specifically for aligning images.

We consider the factorization and classification techniques specifically since they are widely used in the modelling of image relation and propose approaches that are insensitive to misalignment.

1.3.2 Images on manifolds

Most of existing relation modelling techniques assume that images lie in Euclidean space. And a few of them consider the underlying manifold structure of image samples, and assume that images are drawn from a manifold. However, many real world data reside on multiple manifolds [16]. For example, for handwritten digits, each digit forms its own manifold in the feature space. Furthermore, for human faces, the faces of the same person lie on the same manifold and different persons are associated with different manifolds. Many previous relation modelling techniques, such as NMF algorithms, do not consider the structure of multiple manifolds and cannot model data on a mixture of manifolds, which may overlap or intersect. For example, the algorithms in [17,18] only consider the situation that all the data samples are drawn from a single manifold. These algorithms create a graph by searching nearest neighbors to preserve the local geometry information: locality or neighborhood. However, the created graph may connect points on different manifolds, which can diffuse information across manifolds and be misleading.

We here consider two cases where the relation resulting from the underlying multiple manifold structure helps improve the performance.

First, we consider doing clustering with NMF that explicitly models the intrinsic geometrical structure of the data on multiple manifolds. The assumption is that data samples are drawn from multiple manifolds, and if one data point can be reconstructed by several neighboring points on the same manifold in the original high dimensional space, it should also be reconstructed in a similar way within the low dimensional subspace by the basis matrix and coefficient matrix. This approach is different from local linear embedding which studies only one manifold in the original space.

Secondly, we extend the same idea to pattern classification, and hope that the underlying multiple manifold structure would help improve the classification performance. Pattern classification is one of the most important problems in machine learning, and support vector machine(SVM) [19, 20] is one of the most successful approaches. It is widely used in many real applications, such as face recognition [21], object detection [22] and text categorization [23]. In many of these applications, SVM is able to achieve very competitive performance compared with other existing approaches. We aim to combine the discriminative power of support vector machine with the merits of sparse representation resulting from the multiple manifold structure.

1.3.3 Nonlinear relation modelling

Because of the large-scale complex data, linear models are sometimes insufficient due to their simplicity. Kernel trick is one of the most popular choices to model nonlinear relation. Despite kernels' success in many real applications, however, traditional kernel methods usually employ all features and instances to produce a kernel matrix with a high chance of incorporating noise and irrelevant features. For example, in object categorization, the background of an image may greatly impair the clustering accuracy. How to reduce or remove the effects of the background or irrelevant features in kernel construction remains an open problem. The choice of the kernel function is critical to the success of the applications. Thus how to find a good kernel becomes a fundamental topic. The kernel functions basically describe the definition of the similarity between data samples. For many applications we conjecture a sparse kernel would benefit the applications if we have some reason to believe that two distant samples should have zero similarity. Take image categorization for examples, an ideal situation would be that the images from different classes result a zero output, at least a small value, of the kernel function; and images from the same class get a reasonably large positive output. However, current standard kernel functions are designed for general types of data, and thus have a lack of this property. A sparse kernel should also help reduce the computation burden can is more robust to noise. This dissertation introduces an effective and simple approach to attack this problem.

However, the powerful nonlinear approaches, including kernel trick, LLE [24] and I-SOMAP [25], are usually computationally more expensive than linear approaches such as PCA and NMF. To enjoy both the efficiency of the linear models and power of nonlinearity, this dissertation introduces a piecewise linear dimension reduction technique with global consistency and smoothness constraint to overcome the restriction of linearity at relatively low cost in the problem of factorization.

1.4 Dissertation Overview

The rest of this dissertation is organized as follows. Chapters 2 and 3 describe algorithms on relation modelling for images with misalignment. In Chapters 4 and 5, algorithms of factorization and classification considering multi-manifold structure are presented. Chapters 6 and 7 describe two algorithms of nonlinear relation modelling: one is based on sparse kernel trick and the other is piecewise linear approach. In Chapter 8, this dissertation is concluded and future directions are discussed.

2 FACTORIZATION FOR MISALIGNED IMAGES

Factorization techniques, including subspace learning and nonnegative matrix factorization, have been powerful tools for many applications in image analysis, such as face recognition and image clustering. The underlying assumption for factorization is that all the images are well aligned. However, this assumption does not usually hold for real world applications due to image noise and other factors. Thus image alignment is of critical importance to the success of factorization algorithms. On the other hand, factorization has been successfully applied to numerous tasks by learning more informative and less noisy representations, which in turn facilitates the alignment process. This chapter proposes a Transformation Invariant Nonnegative Matrix Factorization algorithm, which aligns and factorizes data simultaneously. Extensive experiments, including image reconstruction, alignment and recognition, on benchmark datasets demonstrate the efficacy of the proposed algorithm.

2.1 Introduction

Factorization techniques have been widely used in many image processing and computer vision tasks. Among these techniques, Nonnegative Matrix Factorization (NMF) [8] has attracted much attention since the values of image data are usually nonnegative in nature. NMF learns a basis matrix and a coefficient matrix, both of which are nonnegative and of low rank, to approximate the data matrix. Numerous variants of NMF methods have been developed for different tasks. Hoyer et al. [26] and Kim et al. [27] propose variants of NMF that explicitly control the sparsity of the resulting basis and coefficient matrices. NMF is later extended to data on a single or multiple manifolds [17, 28]. SemiNMF and convexNMF are proposed in [29] to further extend the applicability of NMF, where only the coefficient matrix is nonnegative and the basis matrix is allowed to contain negative

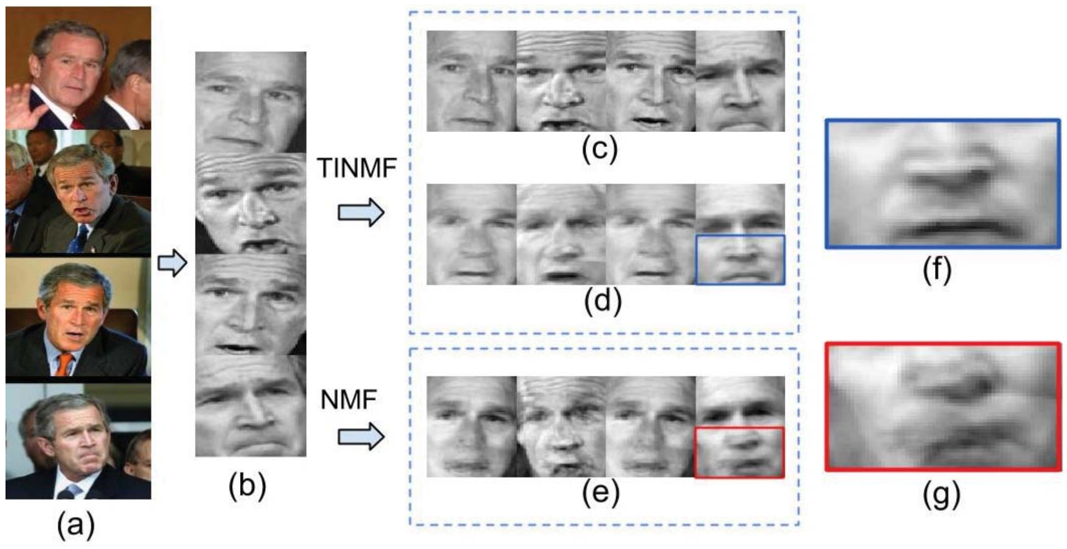


Figure 2.1.: Factorization and alignment of face images. (a) Examples of misaligned celebrity face images from LFW [36]. (b) Detected faces as input. (c)-(d) Images aligned and reconstructed by TINMF. (e) Images reconstructed by NMF. (f)-(g) Cropped reconstruction results by TINMF and NMF.

values. Much effort has been made in recent years for various aspects of NMF including modelling, application and optimization [30–35].

However, performance of most, if not all, existing variants of NMF depends heavily on good alignment of the input samples. Unfortunately, this assumption does not usually hold in many real world applications [9], especially when the raw pixels are treated as features. Real image samples, such as face images, are usually with various poses and/or scales and misaligned due to many uncontrollable factors and noise. Figure 2.1(a) shows some face images in the LFW dataset [36]. The face images here are not well aligned, and so are most images on the Internet. Thus, alignment is a critical problem for the following factorization and other processing processes.

Learned-Miller [10] proposes a congealing algorithm to align images by minimizing the sum of entropy of pixel values at different pixel locations. Alignment of images can also be achieved by other approaches, such as minimizing the sum of squared difference of all pairs [11] or the approximated rank [12]. Huang et al. [9] propose to learn the sparse representation while implicitly aligning the images by inferring the coefficients of the first derivatives. Eggert et al. [13] assume input vectors that are translationally shifted versions of each other, and include a shift-invariant representations into the NMF formulation. Bar et al. [14] develop a scale and rotation invariant NMF algorithm by transforming the images into log-polar coordinate system. He et al. [15] propose to learn a subspace specifically for aligning images to improve accuracy.

However all the existing NMF algorithms either do not address the misalignment problem or only handle the simple cases (e.g., translation), and thus do not perform well when image data contains large misalignment. To address this important issue, we propose to align and factorize images simultaneously. Thus, not only the resulting factorization is insensitive to sample misalignment, but also better alignment is generated by the new representation induced by factorization. We would like to answer the following questions: 1) Does aligning images help learn more informative basis in NMF? 2) Can more accurate alignment be generated when it is coupled with factorization? 3) Is it better to simultaneously align and factorize image data or separately?

2.2 Review of Nonnegative Matrix Factorization

In this section, we first briefly review the conventional NMF algorithm with squared loss objective function.

A set of n samples of m-dimension forms a nonnegative matrix $X \in \mathbb{R}_+^{m\times n}$. An NMF algorithm factorizes it into two nonnegative low rank matrices W and H by minimizing the following objective function:

$$O = ||X - WH||_F^2, \tag{2.1}$$

where $W \in \mathbb{R}_+^{m\times p}$, $H \in \mathbb{R}_+^{p\times n}$. p is usually much smaller than both m and n, and it controls the ranks of matrices W and H.

Although the above objective function is nonconvex with respect to W and H jointly, it is convex with respect to either W or H. An alternative minimization algorithm for optimizing this objective function is to repeat the following two update rules [8]:

$$W_{ij} = W_{ij}\frac{(XH^\top)_{ij}}{(WHH^\top)_{ij}}, \tag{2.2}$$

$$H_{ij} = H_{ij}\frac{(W^\top X)_{ij}}{(W^\top WH)_{ij}}. \tag{2.3}$$

2.3 Transformation Invariant Nonnegative Matrix Factorization

We describe our proposed Transformation Invariant Nonnegative Matrix Factorization (TINMF), which simultaneously factorizes and aligns images for real-world applications.

Given a set of images, $\mathcal{I}_n = \{I_i \in \mathbb{R}_+^{w\times h} : 1 \leq i \leq n\}$, each image is stacked as a vector by the function $vec : \mathbb{R}_+^{w\times h} \to \mathbb{R}_+^m$, where $m = w \times h$. Let $X_{.i}$ denote the vector obtained by vectorizing I_i, i.e. $X_{.i} = vec(I_i)$. We define $X \doteq [X_{.1}, X_{.2}, \ldots, X_{.n}] \in \mathbb{R}_+^{m\times n}$.

To align image I_i, an optimal transformation τ_i is sought in a Lie group $\mathcal{G}$. For implementation, a g-dimensional vector is used to represent τ_i, and $\tau = [\tau_1, \tau_2, \ldots, \tau_n]$ is a matrix of size $g \times n$. With slight abuse of notation, we use τ to denote the set of $\{\tau_i : 1 \leq i \leq n\}$, and $X \circ \tau$ to denote $[vec(I_1 \circ \tau_1), vec(I_2 \circ \tau_2), \ldots, vec(I_n \circ \tau_n)]$.

Given the matrix X formed from image samples, which are potentially misaligned, we are interested in not only the factorization but also the optimal transformation τ. The underlying assumption is that an optimal transformation τ^* can be applied to image set such that the resulting images $X \circ \tau^*$ are well aligned, and the well aligned images have low fitting error while being factorized, i.e., $\min\limits_{W,H} \; \|X \circ \tau^* - WH\|_F$ is small. Thus, the optimal τ can be sought as the transformation that will result in minimal fitting error when the matrix is factorized, i.e. $\tau^* = \operatorname*{argmin}\limits_{\tau}\min\limits_{W,H} \; \|X \circ \tau - WH\|_F$. However, the factorization (W, H) also depends on the alignment τ. Thus, to jointly consider both the factorization and alignment, the objective function can be formulated as

$$\begin{aligned}(W^*, H^*, \tau^*) = &\arg\min_{W,H,\tau} \; \|X \circ \tau - WH\|_F^2, \\ &s.t.\, W \in \mathbb{R}_+^{m\times p}, H \in \mathbb{R}_+^{p\times n}, \tau \in \mathcal{G}.\end{aligned} \tag{2.4}$$

Unfortunately, minimizing the above objective function will result in trivial solution of τ. For example, the τ that transforms all images into single pixels will always result in perfect fitting with zero loss, since $X \circ \tau \in \mathbb{R}_+^{1\times n}$ can be easily fit with zero loss. To avoid this case, the transformation τ should be regularized, so the objective function is reformulated as

$$\begin{aligned}(W^*, H^*, \tau^*) = &\arg\min_{W,H,\tau} \; \|X \circ \tau - WH\|_F^2 + \gamma\, r(\tau), \\ &s.t.\, W \in \mathbb{R}_+^{m\times p}, H \in \mathbb{R}_+^{p\times n}, \tau \in \mathcal{G},\end{aligned} \tag{2.5}$$

where $r(\tau)$ is a regularizer on the transformation τ. Assuming the initial transformation is referred to as τ^o, the regularizer can be defined as $r(\tau) = \sum_i \|\tau_i - \tau_i^o\|_2^2$. The regularizer helps prevent significant trivial transformations.

Optimization

In the objective function, the learning of transformation τ is coupled with the learning of W and H. To optimize this nonconvex objective function, we carry out alternative optimization by repeating these three steps: 1) fix τ and H, update W; 2) fix τ and W, update H; 3) fix W and H, update τ. Obviously, under this interleaving optimization scheme, the reconstruction WH will guide the learning of τ, and in return the current τ will also help learn more informative basis W and representation H.

The optimization process with respect to W or H is straightforward since it is the same as the procedure in standard NMF except that the data X should be substituted by $X \circ \tau$. Thus we here focus on the optimization with respect to τ.

The objective function is not linear or quadratic with with respect to τ, and it is not easy to solve it directly. Instead, we approximate the function and solve it iteratively. When there is a small update $\Delta\tau$ on τ, the updated matrix $X \circ (\tau + \Delta\tau)$ can be approximated by linearization:

$$X \circ (\tau + \Delta\tau) \approx X \circ \tau + \sum_{i=1}^{n} J_i \Delta\tau_i \epsilon_i^\top, \tag{2.6}$$

where J_i is the Jacobian of the i-th image with respect to the transformation τ_i, and ϵ_i is the standard basis for $\mathbb{R}^n$. The loss induced by a single sample is

$$||X_{.i} \circ \tau_i - WH_{.i}||_F^2 + \gamma \left\|\tau_i - \tau_i^o\right\|_2^2, \tag{2.7}$$

where $H_{.i}$ denotes the i-th column of coefficient matrix H.

To optimize with respect to τ_i , i.e., $\arg\min_{\tau_i} ||X_{.i} \circ \tau_i - WH_{.i}||_F^2 + \gamma \left\|\tau_i - \tau_i^o\right\|_2^2$, we solve the following optimization problem with respect to $\Delta\tau_i$:

$$\min_{\Delta\tau_i} ||X_{.i} \circ (\tau_i + \Delta\tau_i) - WH_{.i}||_F^2 + \gamma \left\|(\tau_i + \Delta\tau_i - \tau_i^o)\right\|_2^2. \tag{2.8}$$

Computing the derivative, we have

$$\begin{aligned} &\frac{\partial ||X_{.i} \circ (\tau_i + \Delta\tau_i) - WH_{.i}||_F^2 + \gamma \left\|(\tau_i + \Delta\tau_i - \tau_i^o)\right\|_2^2}{\partial \Delta\tau_i} \\ =&2J_i^\top (X_{.i} \circ \tau_i - WH_{.i}) + 2J_i^\top J_i \Delta\tau_i \\ &+ \gamma(2\Delta\tau_i + 2\tau_i - 2\tau_i^o). \end{aligned} \tag{2.9}$$

Setting the derivative equal to zero, we have the following solution:

$$\Delta\tau_i = [J_i^\top J_i + \gamma I]^{-1} [J_i^\top (WH_{.i} - X_{.i} \circ \tau_i) - \gamma(\tau_i - \tau_i^o)]. \tag{2.10}$$

Algorithm 1 Naive optimization scheme for TINMF

```
Require: Images $\mathcal{I}_n = \{I_i \in \mathbb{R}_+^{w\times h} : 1 \le i \le n\}$, initial transformation $\tau = \tau^o$
 1: Randomly initialize $W$ and $H$ to positive matrices.
 2: Form data matrix $X = [vec(I_1 \circ \tau_1), vec(I_2 \circ \tau_2), \ldots, vec(I_n \circ \tau_n)]$ and normalize $X$
    column-wisely.
 3: while not_converge do
 4:     Update $W$: $W_{ij} = W_{ij}\frac{(XH^\top)_{ij}}{(WHH^\top)_{ij}}$
 5:     Update $H$: $H_{ij} = H_{ij}\frac{(W^\top X)_{ij}}{(W^\top WH)_{ij}}$
 6:     Compute $\Delta\tau$ according to (2.11)
 7:     Update $\tau$: $\tau = \tau + \Delta\tau$
 8:     Update data matrix according to $\tau$: $X = [vec(I_1 \circ \tau_1), vec(I_2 \circ \tau_2), \ldots, vec(I_n \circ \tau_n)]$,
        and normalize $X$ column-wisely.
 9: end while
10: return $\tau, W, H$.
```

Accordingly, we have

$$\begin{aligned}\Delta\tau = &\sum_i [J_i^\top J_i + \gamma I]^{-1}[J_i^\top(WH - X\circ\tau) \\ &- \gamma(\tau - \tau^o)]\epsilon_i\epsilon_i^\top.\end{aligned} \tag{2.11}$$

When $\gamma = 0$, the rules can be simplified as follows:

$$\begin{aligned}\Delta\tau_i &= J_i^+(WH_{.i} - X_{.i}\circ\tau_i), \\ \Delta\tau &= \sum_i J_i^+(WH - X\circ\tau)\epsilon_i\epsilon_i^\top,\end{aligned} \tag{2.12}$$

where J_i^+ is the pseudo inverse of the Jacobian J_i.

Then τ_i can be updated as $\tau_i = \tau_i + \Delta\tau_i$, and this can be repeated to achieve an optimal τ. However, it is not necessary to find an optimal τ in the middle of optimization since it suffices that each step of the alternative optimization decreases the objective value.

Algorithm 2 Efficient optimization scheme for TINMF

```
Require: Images $\mathcal{I}_n = \{I_i \in \mathbb{R}_+^{w\times h} : 1 \le i \le n\}$, initial transformation $\tau = \tau^o$, $maxIter$
 1: Randomly initialize $W$ and $H$ to positive matrices.
 2: Form data matrix $X = [vec(I_1 \circ \tau_1), vec(I_2 \circ \tau_2), \dots, vec(I_n \circ \tau_n)]$ and normalize $X$
    column-wisely.
 3: while not_converge do
 4:     for $i = 1$;$i \le maxIter$;$i$++ do
 5:         Update $W$: $W_{ij} = W_{ij}\frac{(XH^\top)_{ij}}{(WHH^\top)_{ij}}$
 6:         Update $H$: $H_{ij} = H_{ij}\frac{(W^\top X)_{ij}}{(W^\top WH)_{ij}}$
 7:     end for
 8:     Compute $\Delta\tau$ according to (2.11)
 9:     Update $\tau$: $\tau = \tau + \Delta\tau$
10:     Update data matrix according to $\tau$: $X = [vec(I_1 \circ \tau_1), vec(I_2 \circ \tau_2), \dots, vec(I_n \circ \tau_n)]$,
        and normalize $X$ column-wisely.
11: end while
12: return $\tau, W, H$.
```

Algorithm

Following the discussion above, we update W, H and τ alternatively until convergence. The overall procedure is summarized in Algorithm 1.

Since each update step decreases the objective function and obviously this objective function has lower bound, e.g., 0, the algorithm is guaranteed to converge. Though, theoretically the update of τ may not cause a decrease in objective function due to the linearization, in practice this does not happen according to our experiments. Moreover, a simple line search can help prevent the increase in objective function value by scaling down the $\Delta\tau$ once it happens.

However, Algorithm 1 is not efficient because updating τ is computationally more expensive than updating W and H. In addition, the decrease of objective function by updating τ is not as significant as the time complexity. Ideally, during each of these three steps, the individual objective function decrease per unit time can be recorded to tune the frequency of different steps adaptively. For simplicity, we propose to update W and H more frequently by a constant factor, and as a result the decrease of objective function per unit time becomes greater. Thus, it speeds up the overall algorithm. This efficient algorithm is listed in Algorithm 2.

2.4 Experiments

We evaluate variants of NMF on several different tasks including image reconstruction, image alignment and face/digit recognition on different benchmark datasets. To show the advantages of the proposed TINMF algorithm, we compare it with two baseline algorithms. One is the standard NMF algorithm, and the other is a two-stage algorithm. First, given one reference image I_{ref}, we align images by learning the optimal transformation τ^* according to:

$$\begin{aligned} \tau^* =& \underset{\tau \in \mathcal{C}}{\operatorname{argmin}} \sum_i \| \frac{vec(I_i \circ \tau_i)}{\|vec(I_i \circ \tau_i)\|_2} - \frac{vec(I_{ref})}{\|vec(I_{ref})\|_2} \|_2^2 \\ & + \gamma \, r(\tau), \end{aligned} \tag{2.13}$$

which is solved by iteratively linearizing the objective function and updating τ. Then, we form data matrix $X = [vec(I_1 \circ \tau_1^*), vec(I_2 \circ \tau_2^*), \ldots, vec(I_n \circ \tau_n^*)]$, normalize X column-wisely, and factorize it by the standard NMF. For notation convenience, we name this baseline approach *aligned NMF*.

2.4.1 Datasets

The proposed TINMF algorithm is evaluated on face datasets and handwritten digit datasets. Specifically, for face images, we use the CMU Multi-PIE [37], ORL and the Labeled Faces in the Wild (LFW) [36] datasets. For handwritten digits, we use the M-NIST [38] and USPS [39] datasets.

In the CMU Multi-PIE dataset, 20 images with different illumination are captured by a single camera from a fixed view point within only 0.7 seconds. Thus, it is reasonable to assume that these 20 images of the same view of the same subject are well aligned to some extent. Thus, we use these well aligned images to evaluate the performance of

TINMF with simulated controllable misalignment. The ORL database has 40 subjects with 10 images for each of them. The images are taken under different lighting conditions at different time, and are with different facial expression, with or without glasses. It is widely evaluated and the images are often assumed to be aligned. However, as the experimental results will show, aside from the proposed TINMF algorithm, existing NMF methods do not perform well even on this dataset with limited alignment errors. The LFW dataset is a set of face photographs collected from the Internet in real-world uncontrolled environments. It has more natural variation than the ORL dataset. The subset of the LFW dataset we use contains photos of 20 celebrities [12], 19 out of them have 35 photos. The results on this dataset show how well our algorithm handles misalignment in real world data.

The MNIST dataset has 70,000 samples of handwritten digits, which have been normalized to the size of 28×28 and centered. The USPS dataset has 11,000 handwritten digits, which are normalized to the size of 16×16.

2.4.2 Settings

The initial transformations τ^o in TINMF for the Multi-PIE and ORL database are gained by using at only the first image of the first subject, and we set $\tau_1^o = \tau_2^o = \cdots = \tau_n^o$. The initial τ^o for the LFW dataset is set by examing each individual image, since the face positions and poses of photos of LFW are more diverse. When we apply the TINMF and aligned NMF to the images of digits, margins of zeros are padded around the 28×28 or 16×16 images. The initial transformation τ^o for these two datasets are able to crop the original images with sizes of 28×28 and 16×16, respectively. For convenience, we restrict transformation τ to the group of affine transformation, thus the transformation τ_i for any image can be represented by a 6 dimensional vector. Accordingly, τ is represented by a matrix of size $6 \times n$, where n is the number of images.

The parameter p in NMF and TINMF, which controls the rank of the resulting matrices, is simply set to 10 unless explicitly stated in the experiments of recognition. Parameter γ in TINMF is set to 0.01, and we do not tune it across different sets of experiments.

2.4.3 Reconstruction of images

In the reconstruction experiments, TINMF and baseline algorithms are used to learn the basis matrix W and the coefficient matrix H, and then original images are reconstructed as WH. $W \in \mathbb{R}_{+}^{m\times p}$ and $H \in \mathbb{R}_{+}^{p\times n}$ are of the same dimension for all the algorithms involved. As more informative basis is likely to result in better reconstruction, we visualize the reconstructed images for qualitative comparisons, and evaluate them quantitatively.

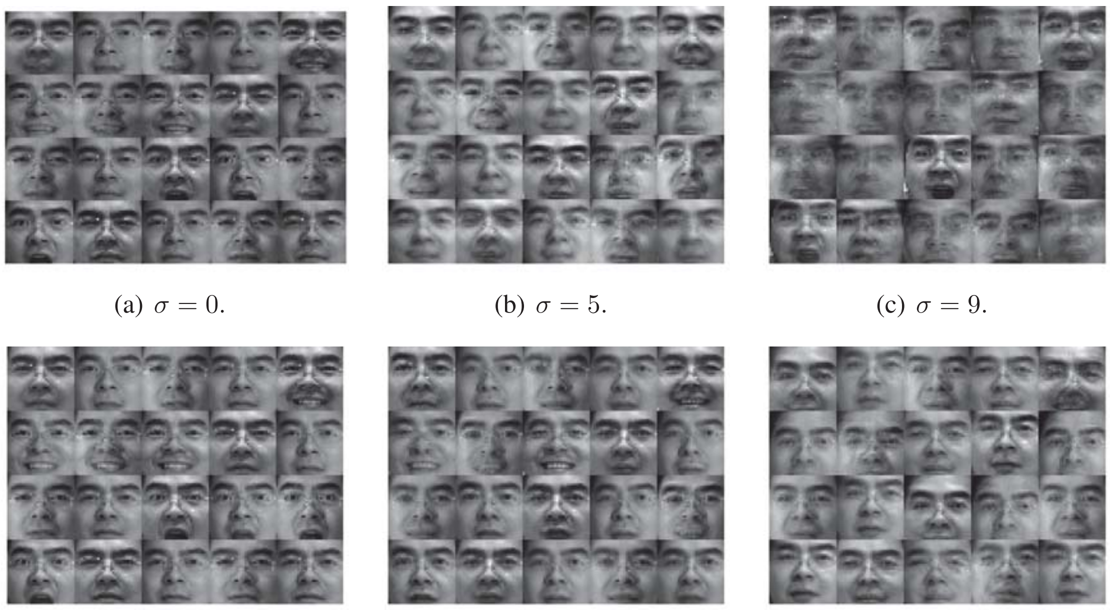
Reconstruction of images with controlled misalignment

To test the ability of our proposed TINMF to handle misalignment, we use the well aligned images from the CMU Multi-PIE dataset. We randomly translate the original images to construct the misaligned input images for TINMF. The amount of translation is controlled by the standard deviation, σ, of the Gaussian noise.

The reconstructed images of a subject with varying noise levels ($\sigma = 0, 5, 9$) by TINMF as well as the baseline algorithm, NMF, are shown in Figure 2.2. When there is no noise ($\sigma = 0$), both NMF and TINMF perform well. However, when there is misalignment, NMF gives low quality reconstruction while TINMF still works well. For example, when $\sigma = 5$, NMF begins to be affected by the misalignment to generate blurred images, while TINMF still has very clear and sharp reconstruction results. When misalignment becomes heavier ($\sigma = 9$), the reconstruction results of TINMF are significantly better than NMF. The experimental results confirm that TINMF is less sensitive to the misalignment than the baseline.

Reconstruction of images with uncontrolled misalignment

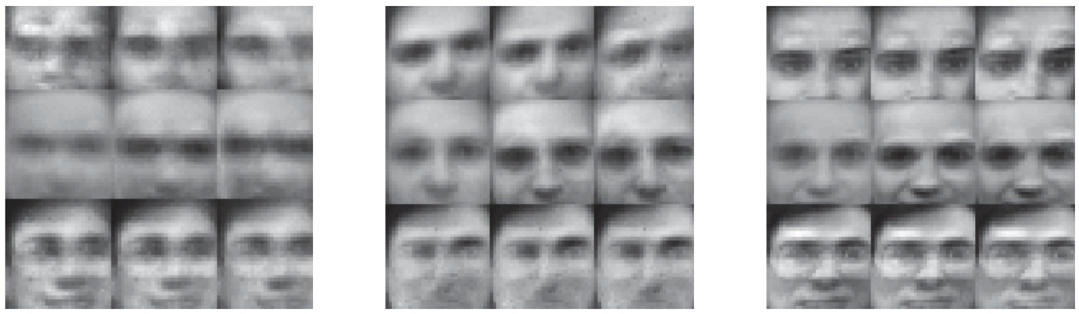
Some reconstructed images in the ORL dataset by TINMF as well as baseline algorithms, NMF and aligned NMF, are shown in Figure 2.3. TINMF gives much clearer reconstruction than both NMF and aligned NMF. To evaluate the results quantitatively, for each subject, the average reconstruction error is defined as $err = \frac{1}{|Sub|}\sum_{I_i \in Sub} \|I_i \circ \tau_i - WH_{.i}\|_2^2$



(a) $\sigma = 0$. (b) $\sigma = 5$. (c) $\sigma = 9$.

(d) $\sigma = 0$. (e) $\sigma = 5$. (f) $\sigma = 9$.

Figure 2.2.: Reconstructed images of a subject in the Multi-PIE dataset. The reconstructed images by NMF are shown in the top row, (a)-(c); the reconstructed images by TINMF are shown in the bottom row, (d)-(f). Different columns account for different levels of misalignment controlled by the value σ. From left to right, the σ values are 0, 5, and 9, respectively.



(a) (b) (c)

Figure 2.3.: ORL reconstruction results. (a) Images reconstructed by NMF. (b) Images reconstructed by aligned NMF. (c) Images reconstructed by TINMF.

for TINMF and aligned NMF; and $err = \frac{1}{|Sub|}\sum_{I_i \in Sub} ||I_i - WH_{.i}||_2^2$ for NMF, where Sub

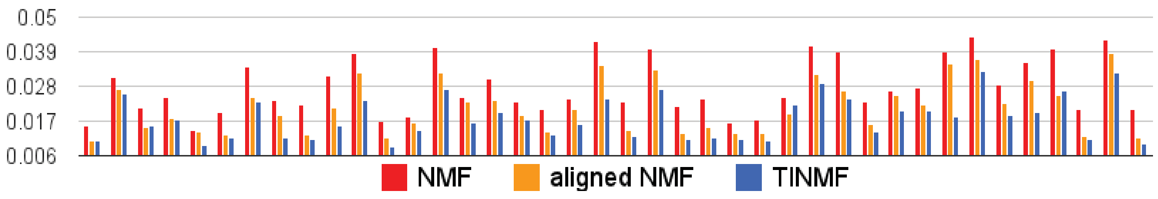


Figure 2.4.: Average reconstruction error of each subject in ORL by NMF, aligned NMF and TINMF.

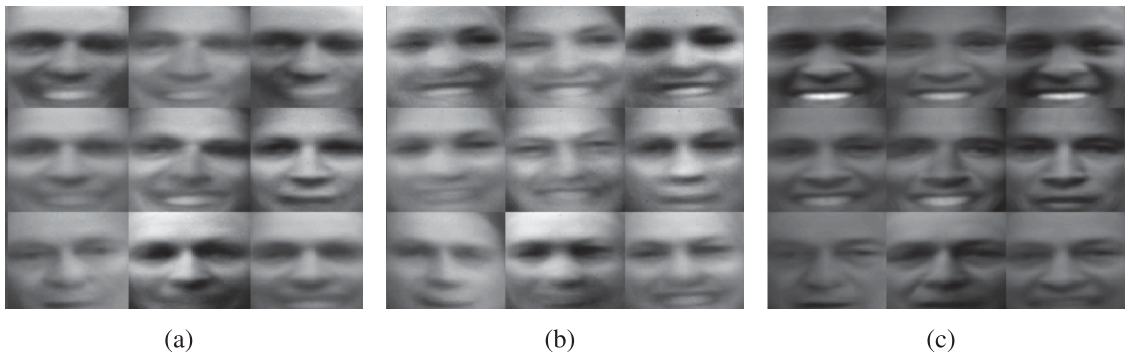


Figure 2.5.: LFW reconstruction results. (a) Images reconstructed by NMF. (b) Images reconstructed by aligned NMF. (c) Images reconstructed by TINMF.

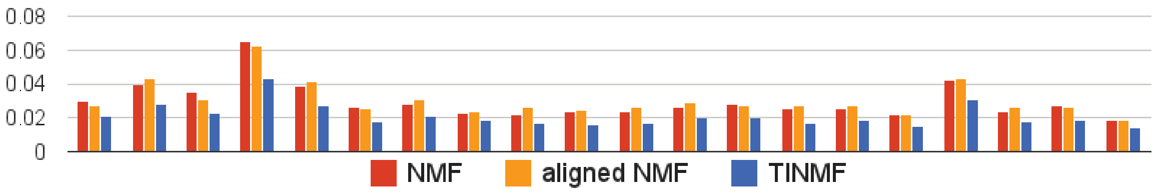


Figure 2.6.: Average reconstruction error of each subject in the LFW dataset by NMF, aligned NMF and TINMF.

is the set of images of this subject and $|Sub|$ is the size. The average reconstruction error for different subjects of the ORL database are shown in Figure 2.4. The average reconstruction errors by TINMF is much smaller than both NMF and aligned NMF, and aligned NMF is better than NMF without alignment. Sample images and quantitative results for the LFW dataset are shown Figure 2.5 and Figure 2.6 respectively. The results also show that TINMF gives much better reconstruction, which has clearer images and smaller reconstruction errors. However, on this dataset with large misalignment, aligned NMF only performs comparably with NMF, which confirms that alignment in the original space is not ideal.

2.4.4 Recognition

Image factorization learns the coefficient matrix, which is referred to as new representation of samples in various applications. In our experiments, we conduct face/digit recognition based on nearest neighbor classifier. The images of each subject are partitioned into two sets, one for training and the other for tests.

In the training phase, all the training images of different subjects are put together, and the factorization algorithms (NMF, aligned NMF, TINMF) learn basis matrix W, and coefficient matrix H. Note the H here is treated as the features of training samples, which will be used later. In the test phase, the basis matrix W is kept fixed as the output W of the training phase, and the test samples are folded in to learn the test coefficient matrix H, which is used as features of test samples. Finally, the nearest neighbor classifier is adopted to classify the test samples based on the Euclidean distance between the features of test and training samples.

Recognition on the ORL dataset

First, we carry out experiments on the ORL dataset. Two sets of experiments are conducted on this dataset. In the first set of experiments, we use all the forty subjects; in the second set of experiments, we use only the first ten subjects. The underlying reason for this design is that alignment will be more critical once there is more diversity and misalignment in pose when we consider more subjects, and theoretically TINMF will outperform NMF more significantly when the aligning process is more demanded.

In both sets of experiments, the first five face images per subject are used as training samples, the other five images per subject are used as test samples. Two different values of

p, which is the dimension of new representation of images after factorization, are considered: 10 and 50. These experiments are repeated ten times because of the randomness and the local optimum resulting from the nonconvexity of the optimization, and the precision reported here is the averaged one over ten runs. The precision is defined as the percentage of correctly predicted testing samples.

Classification results (averaged precision and standard deviation) of ORL database are presented in Table 2.1. For experiments on forty subjects, we see that the precision gained by TINMF is significantly better than both NMF and aligned NMF. When $p = 10$, TINMF has a precision of 66.70%, which is 8.00% higher than NMF; and when $p = 50$, TINMF has 66.79%, which is 6.76% higher than NMF. We note that, when p increases from 10 to 50, the precision of TINMF stays the same, but is still much better than the others. This means, with small amount of basis vectors, TINMF is able to describe the ORL data well enough, while the other algorithms need more. It is intuitive since TINMF learns the new representation and aligns images simultaneously, thus the variation of pose, scale, shift is already considered by τ and the learned representation can be more compact. When it comes with the experiments on ten subjects of the ORL database, there should be less variation of pose and intuitively TINMF will achieve better results than baselines, but the gain may not be as large, as shown in the experimental results.

TINMF performs much better than NMF, which is due to the explicit consideration of transformation when learning the new representation. However, note that the ORL dataset is widely evaluated and the images are usually assumed to be aligned. And, unfortunately, the experimental results have shown that, existing NMF method does not perform well even on this dataset with limited alignment errors, which necessitates the TINMF.

Recognition on the LFW dataset

Then algorithms are also tested on the task of face recognition on a subset of the LFW dataset [12, 37], which is more diverse in pose than ORL. Similarly, we use two sets of experiments with first one using all the twenty subjects while the latter using only the first ten subjects.

In both sets of experiments, the first ten images of all the subjects involved are used as training samples, and the rest are used as test samples. Here three different values of p are considered: 10, 50, and 100, since the face images here are more diverse than the ones in the ORL database. Also, the experiments are repeated by 10 times, and the average precision and standard deviation are reported.

Table 2.2 shows, under any value of p, TINMF significantly outperforms the baseline algorithms consistently for both sets of experiments on ten and twenty subjects. For the experiments with twenty subjects, when $p = 10$, the precision of TINMF is about 3% higher than baselines; when $p = 50$ or $p = 100$, TINMF has up to 6% higher precision. For the experiments on ten subjects of the LFW dataset, TINMF outperforms the baselines consistently again. For example, when $p = 50$, TINMF achieves a precision of 40.04%, while both baselines are less than 35%. These results show that our algorithm handles misalignment in real world well.

Interestingly, for the ORL database, TINMF shows better performance in the set of experiments with forty subjects than the one with ten subjects. That is because forty subjects bring more variation in pose, and transformation estimation is thus more desired there. However, for the experiments on the LFW dataset, TINMF shows nearly equal performance for both sets of experiments with ten subjects and twenty subjects. A reasonable explanation is that the diversity in poses with ten subjects of the LFW dataset is already saturated, and more subjects added will not increase much of the diversity in the dataset.

Table 2.1: Recognition precision on the ORL dataset ($\%$).

	10 subjects		40 subjects	
Methods	p = 10	p = 50	p = 10	p = 50
NMF	84.20 ±3.46	86.10±3.40	58.70 ±2.82	61.03±3.46
aligned NMF	78.80 ±3.29	78.40±2.72	54.35 ±2.58	58.57±4.84
TINMF	**85.80 ±2.39**	**86.70±2.77**	**66.70 ±1.25**	**66.79±1.64**

Table 2.2: Recognition precision on the LFW dataset ($\%$).

	10 subjects			20 subjects		
Methods	p = 10	p = 50	p = 100	p = 10	p = 50	p = 100
NMF	29.96 ± 3.01	32.53± 3.65	32.55± 3.18	15.82 ± 1.24	18.91± 3.54	19.97± 3.32
aligned NMF	33.07 ± 2.43	34.65± 3.18	33.92± 2.99	15.51 ± 1.66	18.27± 3.26	18.75± 3.14
TINMF	**36.15 ± 4.50**	**40.04± 5.28**	**40.17± 4.42**	**18.65 ± 1.24**	**24.96± 6.60**	**25.95± 5.61**

Recognition on handwritten digit datasets

In this section, we show more recognition results on handwritten digit datasets: MNIST and USPS. In each experiment, 2,000 samples are randomly selected without replacement, from which 1,000 of them are treated as training set, and the other samples are for tests. All the experiments are repeated 10 times, and the average precision and standard deviation are reported.

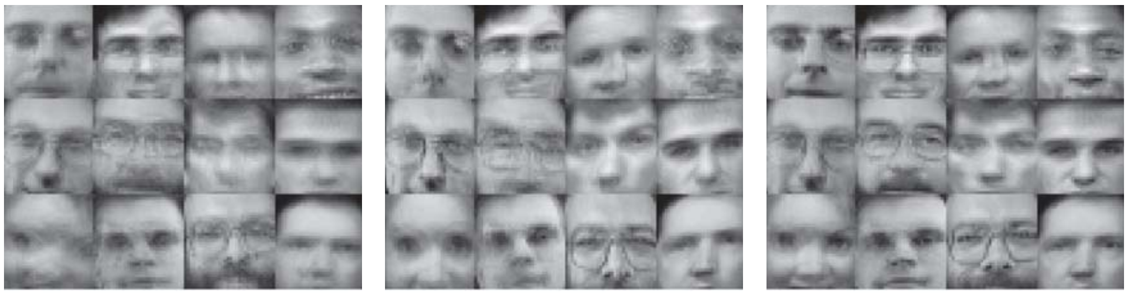
The results on the MNIST and USPS datasets are shown in Table 2.3. With different values of p, TINMF outperforms the baseline algorithms significantly and consistently. For MNIST, when $p = 10$, TINMF is about up to 8% higher than NMF; when $p = 50$ or $p = 100$, TINMF is still significantly better than the baselines. This means when compact low-dimensional representation is preferred, TINMF is a much better choice than NMF or aligned NMF. The lower the dimension is, the more advantage the TINMF has. In other words, with representation of the same dimension, TINMF is able to handle more diverse data. The experimental results on USPS dataset also show significant improvement in classification precision of TINMF over both NMF and aligned NMF. For example, when $p = 10$, the precision of TINMF is 6% higher than NMF.

2.4.5 Aligning images

For the baseline aligned NMF and the proposed TINMF, the optimal τ is inferred before and during the factorization process, respectively. To show whether the inferred τ handles the misalignment successfully, the average aligned images are visualized. Specif-

Table 2.3: Recognition precision on handwritten digit datasets ($\%$).

	MNIST			USPS		
Methods	p = 10	p = 50	p = 100	p = 10	p = 50	p = 100
NMF	76.66 $\pm$ 1.33	81.72$\pm$ 5.32	83.11$\pm$ 4.79	80.45 $\pm$ 1.29	83.91$\pm$ 3.79	84.38$\pm$ 3.18
aligned NMF	78.62 $\pm$ 3.37	82.50$\pm$ 5.02	83.09$\pm$ 4.49	78.36 $\pm$ 10.92	82.26$\pm$ 8.66	83.57$\pm$ 7.34
TINMF	**84.59 $\pm$ 1.69**	**85.04$\pm$ 1.52**	**84.85$\pm$ 1.57**	**86.92 $\pm$ 1.13**	**86.73$\pm$ 1.06**	**86.56$\pm$ 1.19**



(a) (b) (c)

Figure 2.7.: Average (transformed) faces of subjects of ORL database. (a) Average faces before alignment. (b) Averaged aligned faces by aligned NMF (alignment is done using the original feature). (c) Average aligned faces by TINMF.

ically, given the estimated transformation τ, the average image of a certain subject $I_{avg} = \sum_{I_i \in Sub} I_i \circ \tau_i / |Sub|$ is visualized, where Sub is the set of images of this subject, and $|Sub|$ is the size. Note if the images are well aligned, then the average image will be clear, otherwise it will be blurry. The average faces of sample subjects of the ORL database are shown in Figure 2.7. Due to the explicit consideration of transformation τ in the learning processing of TINMF, the resulting average faces by TINMF are much less blurry than the ones from NMF and aligned NMF.

3 CLASSIFICATION FOR MISALIGNED IMAGES

3.1 Introduction

As a fundamental problem in computer vision, image classification has attracted a lot of attention in recent years [40, 41]. The performance of classifier strongly depends on the training samples in both the amount and the quality. Though, in the past decades, the increasing popularity of smart phones and consumer camera has led to a dramatic increase in the amount of visual data online, the accurately labeled data are still expensive to collect in many applications. Further, many image data are of low quality. Specifically, a lot of images are not well aligned even though they are labeled. For example, view images of the same 3D object are usually not perfectly aligned. The misalignment makes classification task difficult in real applications.

For pattern classification, support vector machine(SVM) [19, 20] is one of the most successful and popular approaches. It is widely used in many real applications, such as face recognition [21], object detection [22] and text categorization [23]. In many of these applications, SVM is able to achieve very competitive performance compared with other existing approaches. SVM learns a data classifier in a fully supervised manner. Given a set of labeled data samples, it aims to find the hyperplane which separates samples with different labels with the largest margin. Later, it is extended to semi-supervised fashion [42]. This semi-supervised version of SVM tries to preserve the neighborhood relation among the unlabeled samples while searching for the separating hyperplane for those labeled samples. In this manner, both the labeled and unlabeled samples contribute to the final solution. Furthermore, maximum margin clustering [43], which can be viewed as an unsupervised counterpart of SVM, tries to find hyperplane which separates the unlabeled samples with largest margin. However, all of these existing variants of SVM strongly rely on the training data, for example, they assume the images are well aligned, which is usually not true in real world. Here we only discuss SVM, though this is not limited to SVM and many computer vision algorithms assume that samples are well aligned [28, 44, 45].

Given limited amount of training samples, how to make better use of that? Is it possible to compensate for the misalignment between training and testing samples without having to look at the testing ones? This dissertation explicitly considers the misalignment between training and testing images in classification task and proposes to compensate for the worst case of possible misalignment without looking at the testing samples by inferring and applying some certain transformations to the training data samples while learning the classifier. The resulted classifiers are shown to have better generalization performance in testing phase.

3.2 Review of Support Vector Machine

In this section, we briefly review the formulation of large margin classifier, i.e. Support Vector Machine(SVM).

Given a set of labeled samples $\{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^m$ is the feature vector of i_{th} sample and y_i is the corresponding label(for binary classification, $y_i \in \{-1, +1\}$), SVM learns a hyperplane $w \in \mathbb{R}^m$ that maximizes the margin between samples.

$$\min_w C \sum_i \max(0, 1 - y_i w^T x_i) + \frac{1}{2}\|w\|^2. \tag{3.1}$$

The first term in (3.1) is the hinge loss, the second term is a regularizer for w, and C is a parameter controlling the tradeoff between the hinge loss and the regularization on w. The explicit bias term is omitted here to simplify the problem while keeping it entirely general since the bias could be easily introduced as an additional coordinate in the data.

3.3 Transformation Invariant Support Vector Machine

In real applications of image classification, the data are usually not perfectly aligned, and the misalignment between training and testing images may greatly impair the performance of classifier. Transformations that exactly compensate the misalignment are able to solve this problem by applying them to samples. Unfortunately, this is impossible because the real misalignment cannot be estimated since testing samples are unknown in the training phase. A realistic way to alleviate this problem is to randomly sample as many transformations as possible to cover all the possible misalignment in the training phase, and each sampled transformation for each training image will generate a transformed sample. However, this blind random sampling scheme is too inefficient and cannot scale up. To alleviate this problem efficiently, the proposed algorithm considers worst cases in training phase. Concretely, the transformations that will make the training samples most difficult to classify are inferred and then applied to the training samples while learning the large margin classifier.

Here we propose the idea of Transformation Invariant Support Vector Machine(TISVM), which is able to infer critical transformations to apply to the existing samples to generate new training samples while learning the large margin classifier simultaneously. Transformations are called to be critical if they are able compensate for the *worst* misalignment between training and testing images.

3.3.1 Simultaneously transform and train

Given a set of images $\mathcal{I}_n = \{I_i \in \mathcal{R}^{w\times h} : 1 \le i \le n\}$, each image is stacked as vector by a function $vec : \mathcal{R}^{w\times h} \to \mathcal{R}^m$, where $m = wh$. Let x_i denote vector obtained by vectorizing I_i, i.e. $x_i = vec(I_i)$. Then we define $X \doteq [x_1, x_2, \ldots, x_n] \in \mathcal{R}^{m\times n}$. And y_i is the label of sample x_i.

For image I_i, a critical transformation τ_i is sought in a certain Lie group $\mathcal{G}$ in the training phase. In implementation, we use a g dimensional vector to represent τ_i. With a little abuse of notation, we use τ to denote the set of $\{\tau_i : 1 \le i \le n\}$, $x_i \circ \tau_i$ to denote $vec(I_i \circ \tau_i)$

and $X \circ \tau$ to denote $\{vec(I_1 \circ \tau_1), vec(I_2 \circ \tau_2), \ldots, vec(I_n \circ \tau_n)\}$, where $I_i \circ \tau_i$ denotes the transformed image generated by applying transformation τ_i to image I_i.

The objective function can be:

$$\min_w C \sum_i \max(0, 1 - y_i w^T x_i \circ \tau_i) + \frac{1}{2}\|w\|^2. \tag{3.2}$$

Note, $x_i \circ \tau_i$ tries to apply a transformation τ_i for each of the training samples so as to generate new samples. Ideally, for negative samples, applying transformation will always lead to new negative samples. We seek transformations that will make the new negative samples as *positive*/confusing as possible, since these transformations are capable of compensating the worst cases of misalignment. For negative samples, a critical transformation is sought by solving this problem $\max_{\tau_i \in \mathcal{C}_i} w^T x_i \circ \tau_i$, where the constraint in $\mathcal{C}_i$ ensures the transformation is slight enough so that the label is still valid after transformation and the $\max$ ensures the resulting samples are confusing and difficult. For the positive samples, small transformations are also applied so that the transformed samples are as confusing as possible. Similarly, each critical transformation is inferred by solving an optimization problem $\min_{\tau_i \in \mathcal{C}_i} w^T x_i \circ \tau_i$.

Accordingly, after balancing the size of positive and negative samples, the formulation is revised to:

$$\begin{aligned} \min_w C(\frac{1}{|\{i'|y_{i'} = -1\}|} \sum_{i \in \{i'|y_{i'}=-1\}} \max(0, 1 + \max_{\tau_i \in \mathcal{C}_i} w^T x_i \circ \tau_i) \\ + \frac{1}{|\{i'|y_{i'} = 1\}|} \sum_{i \in \{i'|y_{i'}=1\}} \max(0, 1 - \min_{\tau_i \in \mathcal{C}_i} w^T x_i \circ \tau_i)) + \frac{1}{2}\|w\|^2. \end{aligned} \tag{3.3}$$

Note, with little abuse of notation mentioned above, here $x_i = x_i \circ \tau_i^o$. In the formulation above, the confusing samples are generated by inferring critical transformations and the large margin classifier is trained simultaneously.

3.3.2 Optimization

In the objective function, the learning of transformation τ is coupled with the learning of w. The problem involves not only solving the w, but also inferring the transformation τ. To optimize this objective function, we do alternative optimization by repeating these two steps: 1, fix τ, update w; 2, fix w, update τ. Obviously, under this interleaving optimization scheme, the separating plane w will guide the learning of τ, and in return the current τ will also help learn more discriminative separating plane w.

Optimize w.r.t. w The optimization problem with respect to w in Equation 3.3 is convex. Note both $\max$ and $-\min$ ensures the final convexity. Because $w^T x_i \circ \tau_i$ is convex with respect to w, then $\max_{\tau_i \in \mathcal{C}_i} w^T x_i \circ \tau_i$ and $-\min_{\tau_i \in \mathcal{C}_i} w^T x_i \circ \tau_i$ are also convex.

If $\{\tau_i : 1 \leq i \leq n\}$ are given, to solve the optimization problem defined above with respect to w, we can easily rely on standard convex optimization techniques. Specifically, here we adopt Quasi-Newton method to solve it.

Optimize w.r.t. τ However, the optimal transformation τ_i depends on the w. Thus, we solve the following maximization and minimization problems for negative and positive samples, respectively:

$$\begin{aligned} &\max_{\tau_i} w^T x_i \circ \tau_i \qquad && and \qquad && \min_{\tau_i} w^T x_i \circ \tau_i \\ &s.t. \quad \tau_i \in \mathcal{C}_i && && s.t. \quad \tau_i \in \mathcal{C}_i \end{aligned} \tag{3.4}$$

For τ_i, there is an initial transformation τ_i^o. So, we define $\mathcal{C}_i$ as the neighborhood of τ_i^o, i.e. $\mathcal{C}_i = \mathcal{N}(\tau_i^o, \epsilon_i)$. Specifically, $\mathcal{C}_i = \{\tau_i | \|\tau_i - \tau_i^o\| \leq \epsilon_i\}$.

Thus, the maximization problem for negative samples becomes(the minimization problem is similar)

$$\begin{aligned} &\max_{\tau_i} w^T x_i \circ \tau_i \\ s.t.\quad &\|\tau_i - \tau_i^o\| \leq \epsilon_i \end{aligned} \tag{3.5}$$

which is equivalent to the following problem

$$\min_{\tau_i} -w^T x_i \circ \tau_i + \frac{1}{2}\lambda_i \|\tau_i - \tau_i^o\|^2 \tag{3.6}$$

Instead of setting parameter ϵ_i, we set λ_i. For simplicity, we set $\lambda = \lambda_1 = \lambda_2 = \cdots = \lambda_n$.

Unfortunately, the operation by τ is nonlinear, and makes the direct optimization difficult. To avoid the difficulty by nonlinearity, we linearize $x_i \circ \tau_i$ by first order approximation. When there is small update $\Delta\tau_i$ on τ_i, the updated sample $x_i \circ (\tau_i + \Delta\tau_i)$ can be approximated by linearization:

$$x_i \circ (\tau_i + \Delta\tau_i) \approx x_i \circ \tau_i + J_i \Delta\tau_i \tag{3.7}$$

where J_i is the Jacobian of the i-th image with respect to the transformation τ_i.

To optimize with respect to τ_i , we solve the following problem, where we optimize with respect to $\Delta\tau_i$ iteratively:

$$\min_{\Delta\tau_i} -w^T (x_i \circ \tau_i + J_i \Delta\tau_i) + \frac{1}{2}\lambda \|\tau_i - \tau_i^o\|^2 \tag{3.8}$$

Compute the derivative, we have

$$\begin{aligned}
&\frac{\partial - w^T(x_i \circ \tau_i + J_i \Delta\tau_i) + \frac{1}{2}\lambda\|\tau_i + \Delta\tau_i - \tau_i^o\|^2}{\partial \Delta\tau_i} \\
=& - J_i^\top w + \frac{1}{2}\lambda \frac{\partial \|\tau_i + \Delta\tau_i - \tau_i^o\|^2}{\partial \Delta\tau_i} \\
=& - J_i^\top w + \lambda(\Delta\tau_i + \tau_i - \tau_i^o)
\end{aligned} \tag{3.9}$$

To set the derivative with respect to $\Delta\tau_i$ to zero, we get the optimal $\Delta\tau_i$ as

$$\Delta\tau_i = \frac{1}{\lambda} J_i^\top w - \tau_i + \tau_i^o \tag{3.10}$$

Similarly, for the minimization problem for each of the positive samples, the optimal $\Delta\tau_i$ is

$$\Delta\tau_i = -\frac{1}{\lambda} J_i^\top w - \tau_i + \tau_i^o \tag{3.11}$$

Once we have the $\Delta\tau = \{\Delta\tau_1, \Delta\tau_2, \ldots, \Delta\tau_n\}$, the transformation τ is updated by

$$\tau_i \leftarrow \tau_i + \Delta\tau_i \tag{3.12}$$

3.3.3 Overall algorithm of TISVM

With the initial input, a SVM classifier is trained on these samples, and then the training samples are made more confusing by tuning the transformation τ for generating new samples, which can be then fed to the learning algorithm to train a new SVM. Each iteration we get new samples which are difficult for the classifier learned from previous iteration.

The newly generated samples are difficult for only the classifier learned from the previous iteration. It may be risky to only use the newly generated samples. Thus, we relax it to a conservative version, which keeps all the difficult samples of different iterations.

Basically, for each iteration, a classifier is trained, and new training samples, which are difficult for the current classifier, are generated and then added to the training data to train new classifier. The overall procedure of TISVM is summarized in Algorithm 3.

Algorithm 3 Transformation invariant support vector machine

```
Require: Images $\mathcal{I}_n = \{I_i \in \mathcal{R}^{w \times h} : 1 \le i \le n\}$, initial transformation $\tau = \tau^o$, and labels $\{y_i : 1 \le i \le n\}$.
 1: Form data samples $S_p = \{x_i \circ \tau_i^o : y_i = 1, 1 \le i \le n\}$ and $S_n = \{x_i \circ \tau_i^o : y_i = -1, 1 \le i \le n\}$, where $x_i \circ \tau_i^o = normalize(vec(I_i \circ \tau_i^o)), i = 1, 2, \dots, n$. With little abuse of notation, $x_i = x_i \circ \tau_i^o$. Initialize $k$ = 0.
 2: while $k \le K$ do
 3:     Train SVM: $w^k = \arg\min_w C(\frac{1}{|S_n|} \sum_{i \in \{i' | x_{i'} \circ \tau_{i'}^k \in S_n\}} \max(0, 1 + w^T x_i \circ \tau_i^k) + \frac{1}{|S_p|} \sum_{i \in \{i' | x_{i'} \circ \tau_{i'}^k \in S_p\}} \max(0, 1 - w^T x_i \circ \tau_i^k)) + \frac{1}{2}\|w\|^2$
 4:     for $i \in \{1, 2, \dots, n\}$ do
 5:         $\Delta\tau_i^k = -y_i \frac{1}{\lambda} J_i^\top w - \tau_i^k + \tau_i^o$
 6:         $\tau_i^{k+1} \leftarrow \tau_i^k + \Delta\tau_i^k$
 7:     end for
 8:     Update data samples: $S_n \leftarrow S_n \bigcup \{x_i \circ \tau_i^{k+1} : y_i = -1, 1 \le i \le n\}$ and $S_p \leftarrow S_p \bigcup \{x_i \circ \tau_i^{k+1} : y_i = 1, 1 \le i \le n\}$, where $x_i \circ \tau_i^{k+1} = normalize(vec(I_i \circ \tau_i^{k+1}))$.
 9:     $k$++
10: end while
11: return $w^K$.
```

3.4 Experiments

In this section, we describe the experimental settings, and show and briefly analyze the results.

3.4.1 Experimental settings

The experiments are conducted on two real data sets. One is USPS handwritten digi data set, and the other is a subset of the Labeled Faces in the Wild(LFW) [12, 36]. Both of them are popularly adopted as benchmark data sets for computer vision, image processing algorithms.

The USPS data set has gray scale images of hand written digits from 0 through 9, each image is of size 16×16, and each digit has 1100 images. The data set is simple, however, samples are still not perfectly aligned, as will be confirmed by our experiments. Sample images are shown in Figure 3.1. In experiments, 200 images are randomly selected as training samples, and 1000 other images are randomly selected as testing samples. For each class i, the samples of the training samples belonging to class i are treated as positive samples; the samples belonging to other classes are treated as negative samples.

The LFW data set is a set of face photographs collected from the Internet and designed for the research problems with face image unconstrained environment. Thus, there is much misalignment in this data set. The subset of LFW contains photos of twenty celebrities, nineteen out of them have thirty-five photos, and the last one, *Barack Obama* has sixteen images. Specifically, these celebrities are *Gloria Macapagal Arroyo, Jennifer Capriati, Laura Bush, Serena Williams, Barack Obama, Ariel Sharon, Arnold Schwarzenegger, Colin Powell, Donald Rumsfeld, George W Bush, Gerhard Schroeder, Hugo Chavez, Jacques Chirac, Jean Chretien, John Ashcroft, Junichiro Koizumi, Lleyton Hewitt, Luiz Inacio Lula da Silva, Tony Blair, Vladimir Putin.* Sample images are listed in Figure 3.2. For each class(say there are k images), 10 images are randomly selected as positive training samples, and the rest $k - 10$ serve as positive testing samples. 10 images from other classes are selected as negative training samples, and other $k - 10$ images from other classes are randomly selected as negative testing samples.

The proposed TISVM is compared with the traditional SVM on these two data sets. For the parameter λ, this can be set by cross validation. We do not tune it with different categories of different data sets, and set λ to 100 for all categories for both data sets.

3.4.2 Experimental results

The results on the data set USPS are shown in Table 3.1. Note that the results reported here are the average precision over ten independent runs. We do this to avoid the randomness induced in the different steps in the experiments, such as the samples selection, and to

Table 3.1: Recognition precision on 10 subjects of USPS dataset.

Methods	SVM	TISVM	Methods	SVM	TISVM
Digit 0	**0.883**	0.873	Digit 5	0.921	**0.933**
Digit 1	0.812	**0.864**	Digit 6	0.925	**0.959**
Digit 2	**0.870**	0.855	Digit 7	0.751	**0.770**
Digit 3	0.873	**0.891**	Digit 8	0.851	**0.874**
Digit 4	**0.868**	0.807	Digit 9	0.884	**0.899**

get statistically meaningful results. The number in bold emphasizes the winner algorithm. From the results we can see for 7 out of the 10 categories of USPS data set, the proposed TISVM beats traditional SVM.

The results on the data set LFW are shown in Table 3.2. From the results on the benchmark data set LFW, we see that in 14 out of 20 categories, TISVM beats SVM. The LFW data set contains face photos taken in uncontrolled environment, thus there are a lot of misalignment in face images of the same subject. The proposed TISVM will infer critical transformation τ to generate informative samples, which alleviates the difficulty results from the misalignment between the training samples and the testing samples, and thus

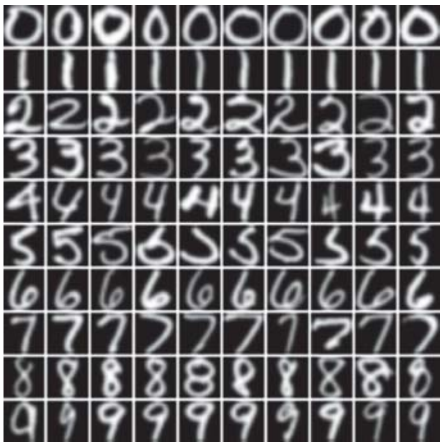


Figure 3.1.: Examples of USPS images.

Figure 3.2.: Examples of LFW images.

Table 3.2: Recognition precision on 20 subjects of LFW dataset.

Methods	SVM	TISVM	Methods	SVM	TISVM
Class 1	0.927	**0.947**	Class 11	0.667	**0.707**
Class 2	0.800	**0.820**	Class 12	0.573	**0.593**
Class 3	**0.927**	0.920	Class 13	0.727	**0.747**
Class 4	**0.853**	0.847	Class 14	0.787	**0.833**
Class 5	**0.944**	0.889	Class 15	0.753	**0.780**
Class 6	**0.753**	0.720	Class 16	0.800	**0.840**
Class 7	0.547	**0.600**	Class 17	0.813	**0.820**
Class 8	0.540	**0.580**	Class 18	**0.747**	0.740
Class 9	0.600	**0.613**	Class 19	0.687	**0.720**
Class 10	0.593	**0.640**	Class 20	**0.700**	0.693

boosts the classification performance, which is shown in the table. Averaged over all the twenty classes, SVM has a precision of 73.69%, while TISVM achieves 75.24%.

4 NONNEGATIVE MATRIX FACTORIZATION CLUSTERING FOR DATA ON MULTIPLE MANIFOLDS

Nonnegative Matrix Factorization (NMF) is a widely used technique in many applications such as clustering. It approximates the nonnegative data in an original high dimensional space with a linear representation in a low dimensional space by using the product of two nonnegative matrices. In many applications with data such as human faces or digits, data often reside on multiple manifolds, which may overlap or intersect. But the traditional N-MF method and other existing variants of NMF do not consider this. This chapter proposes a novel clustering algorithm that explicitly models the intrinsic geometrical structure of the data on multiple manifolds with NMF. The idea of the proposed algorithm is that a data point generated by several neighboring points on a specific manifold in the original space should be constructed in a similar way in the low dimensional subspace. In this way, the structure information on manifolds are used to help NMF. A set of experimental results on two real world datasets demonstrate the advantage of the proposed algorithm.

4.1 Introduction

Nonnegative Matrix Factorization (NMF) has been applied in many applications such as clustering and classification. It provides a linear representation of nonnegative data in high dimensional space with the product of two nonnegative matrices as a basis matrix and a coefficient matrix. NMF has received substantial attention due to its theoretical interpretation and desired performance.

Several variants of NMF has been proposed recently. Sparseness constraints have been incorporated into NMF to obtain sparse solutions [27,46]. Discriminative NMF algorithms have been proposed in [47,48] to maximize the between-class distance and minimize the within-class distance when learning the basis and coefficient matrices. Some recent research work suggest data of many applications in a high dimensional Euclidean space are usually embedded in a low dimensional manifold [24,49]. To explore the local structure on the low dimensional manifold, research work in [17,18] have proposed Locality Preserving NMF and Neighbourhood Preserving NMF, which add constraints between a point and its neighboring one(s).

However, many real world data reside on multiple manifolds [16]. For example, for handwritten digits, each digit forms its own manifold in the feature space. Furthermore, for human faces, the faces of the same person lie on the same manifold and different persons are associated with different manifolds. All existing NMF algorithms do not consider the geometry structure of multiple manifold and cannot model data on a mixture of manifolds, which may overlap or intersect. In particular, the algorithms in [17, 18] only consider the situation that all the data samples are drawn from a single manifold. These algorithms create a graph by searching nearest neighbor(s) to preserve the local geometry information: locality or neighborhood. However, the created graph may connect points on different manifolds, which can diffuse information across manifolds and be misleading.

This chapter proposes a novel clustering algorithm with NMF that explicitly models the intrinsic geometrical structure of the data on multiple manifolds. The assumption is that data samples are drawn from multiple manifolds, and if one data point can be reconstructed by several neighboring points on the same manifold in the original high dimensional space, it should also be reconstructed in a similar way within the low dimensional subspace by the basis matrix and coefficient matrix. This approach is different from local linear embedding which only studies one manifold in the original space. Multiplicative updating rules are derived for the proposed algorithm with guaranteed convergence.

The proposed NMF clustering algorithm on multiple manifolds has been evaluated on two real world datasets. It has been shown to generate more accurate and robust results than the K-means and two variants of existing NMF algorithms. Furthermore, it has been shown that the new algorithm can learn better representation for data in different classes than the traditional NMF method.

4.2 Review of Nonnegative Matrix Factorization

Given a nonnegative matrix $X \in {\mathbb{R}_+}^{m \times n}$, each column of X is a data sample. The NMF algorithm aims to approximate this matrix by the product of two nonnegative matrices $W \in {\mathbb{R}_+}^{m \times k}$ and $H \in {\mathbb{R}_+}^{k \times n}$. To achieve this, the following objective function is minimized:

$$\begin{aligned} O &= ||X - WH||_F^2 \\ s.t.\ \ W &\geq 0, H \geq 0 \end{aligned} \tag{4.1}$$

where $||.||_F$ denotes the Frobenius norm.

The following iterative multiplicative updating algorithm is proposed in [8] to minimize the objective function:

$$W_{ij} = W_{ij} \frac{(XH^T)_{ij}}{(WHH^T)_{ij}} \tag{4.2}$$

$$H_{ij} = H_{ij} \frac{(W^T X)_{ij}}{(W^T W H)_{ij}} \tag{4.3}$$

4.3 Nonnegative Matrix Factorization Clustering on Multiple Manifolds

This section presents the formulation of the proposed Nonnegative Matrix Factorization on Multiple Manifolds(MM-NMF for short). It then describes the optimization algorithm.

4.3.1 Formulation

Given a nonnegative matrix X, each column of which is a data sample. In our target applications with data such as human faces and digits, our assumption is that data samples are drawn from a mixture of manifolds. We use a product of two nonnegative matrices W and H to approximate X while taking the multiple manifold structure information into consideration. The matrix W can be viewed as the basis, while the matrix H can be treated as the coefficients.

Considering that data samples are drawn from different manifolds, the matrix W represent bases for different manifolds, and a data sample $X_{.i}$ only belongs to a manifold (as long as it is not drawn at the intersection). We denote the manifold from which $X_{.i}$ is drawn as M_i. Ideally, there should be a subset of columns of W associated with the manifold M_i. So the corresponding coefficient vector of this sample $H_{.i}$ should have nonzero values only for the entries which correspond to the subset of columns of W associated with the manifold M_i. Thus, the coefficient matrix H should be naturally sparse.

Another important goal is to encode the geometrical information of multiple manifolds. When there are more than one manifolds, we cannot create a graph which directly connects the neighboring samples according to Euclidean distance, since samples close to each other may belong to different manifolds, especially near the intersection of different manifolds.

Based on the above discussion, our MM-NMF algorithm on multiple manifolds should be equipped with two properties: 1) the coefficient matrix H is sparse. In other words, the representation of the samples in the new space is sparse; 2) the local geometrical information on each manifold is preserved. In the following part, we will describe how we formulate MM-NMF with these two desired properties.

4.3.2 Sparse representation in output space

The traditional NMF will learn part-based representation due to its nonnegativity constraint. This means that the data representation in the output space spanned by W is sparse.Our algorithm also inherits the nonnegativity constraint, which will introduce some sparseness.

However, the sparseness introduced by nonnegativity may not be enough and is difficult to control [46]. Some prior research made the matrix sparse by adding an extra sparseness constraint which is usually related with the L1 norm minimization technique [50]. We utilize the method in [27] , which is denoted as SNMF, to make the coefficient matrix H sparse.

The objective function of SNMF is defined as follows:

$$O_{SNMF} = ||X - WH||_F^2 + \zeta \|W\|_F^2 + \lambda \sum_j \|H_{.j}\|_1^2 \tag{4.4}$$

where $H_{.j}$ is the jth column of H. The term $\lambda \sum_j \|H_{.j}\|_1^2$ encourages the sparsity, and $\zeta \|W\|_F^2$ is used to control the scale of matrix W. Parameter λ controls the desired sparsity, and ζ is simply set as the maximum value of X.

Since the nonnegativity constraint automatically introduces some degree of sparseness, we will study two cases with and without the extra sparseness regularizer $\lambda \sum_j \|H_{.j}\|_1^2$ in this chapter.

4.3.3 Mining intrinsic geometry on individual manifolds

This section targets on the second property, which is to preserve the local geometrical information on each manifold in the matrix factorization. We try to preserve the neighborhood relation on each manifold. Note that the neighborhood relation is defined on the manifold rather than in the Euclidean space. This local geometry information on each manifold will guide the formulation of sparseness, which is similar with joint sparsity constraint [51].

To achieve the goal, we assume there are enough data samples so that any data sample can be well approximated by a linear combination of neighboring data samples on the same manifold, since the manifold is usually smooth [52].

Now we describe how to explore the intrinsic geometry in the data matrix X with size of $M \times N$. Let $X_{.i}$ be the i_{th} sample, which is under consideration now. We want to identify its neighbors on the same manifold rather than the entire Euclidean space. As there are enough samples and the manifold is smooth, the data point can be well approximated by a linear combination of a few nearby samples on the same manifold. Thus, it has a sparse representation over the entire data matrix X. To identify the set of samples that can approximate $X_{.i}$, we use the sparsest linear combination which can approximate $X_{.i}$ by the L1 norm minimization technique [50]. We obtain a sparse structure matrix S from the equation of $X = XS$, where the diagonal elements of S are 0. This means any sample can be expressed by several other samples on the same manifold. We construct the S as follows.

For any $i = 1, ..., N$

$$\begin{aligned} &\min_{S_{.i}} ||S_{.i}||_1 \\ s.t. \quad &X_{.i} = XS_{.i} \\ &S_{ii} = 0 \end{aligned} \tag{4.5}$$

There may be some noise in real applications and the equality constraint above may not hold, we relax it to the following equation:

$$\begin{aligned} &\min_{S_{.i}} ||S_{.i}||_1 \\ s.t. \quad &\|X_{.i} - XS_{.i}\|_2 < \epsilon \\ &S_{ii} = 0 \end{aligned} \tag{4.6}$$

Ideally, the nonzero entries in the vector $S_{.i}$ correspond to the samples which lie on the same low dimensional manifold as $X_{.i}$. On the other hand, the nearest samples in Euclidean space from another manifold may not appear in the nonzero entries. ϵ controls the noise energy, and is set to 0.01 here.

4.3.4 MM-NMF objective function

We preserve the geometry relation represented by S in the matrix factorization. When the matrix is factorized, we try to preserve geometry constraint from S for H. This can be gained by minimizing

$$\begin{aligned}
&\sum_i ||H_{.i} - HS_{.i}||_2 \\
&= ||H - HS||_F \\
&= ||H(I - S)||_F \\
&= tr(H(I - S)(I - S)^T H^T) \\
&= tr(HGH^T)
\end{aligned} \tag{4.7}$$

where $I \in \mathbb{R}_+{}^{n \times n}$, and $G = (I - S)(I - S)^T$.

Considering both of the two properties we want to engage: 1, sparseness; 2, local structure preservation on each manifold, the objective function of MM-NMF is defined as:

$$\begin{aligned}
O_{MM-NMF} = ||X - WH||_F^2 + \zeta \|W\|_F^2 + \lambda \sum_j \|H_{.j}\|_1^2 \\
+ \eta tr(HGH^T)
\end{aligned} \tag{4.8}$$

When minimizing the objective function above, we should add constraints that the W and H be nonnegative. The first term is the square fitting error, the second term controls the energy of W, the third term encourages the sparseness, and the last term is to preserve the local geometry structure on each manifold.

Now consider a special case of MM-NMF, where there is no sparseness regularizer($\lambda = 0$). This means we only rely on the nonnegativity constraint to engage the H with first property: sparseness. We name this special case as MM-NMF2. The objective function is:

$$O_{MM-NMF2} = ||X - WH||_F^2 + \zeta \|W\|_F^2 + \eta tr(HGH^T) \tag{4.9}$$

4.3.5 Optimization

Here we only consider how to optimize the objective function of MM-NMF, since MM-NMF2 is a special of case of MM-NMF.

Since O_{MM-NMF} is not convex with W and H jointly, it is difficult to find the global minimum for O_{MM-NMF}. Instead, we aim to find a local minimum for O_{MM-NMF} by iteratively updating W and H in a similar way with the work [8] for NMF.

Update W

Given H, we update W to decrease the value of objective function.In the following equation we follow [27] to reformulate the objective function for computational convenience.

$$\begin{aligned} W =& arg \min_{W \geq 0} ||X - WH||_F^2 + \zeta \|W\|_F^2 + \lambda \sum_j \|H_{.j}\|_1^2 \\ &+ \eta tr(HGH^T) \\ =& arg \min_{W \geq 0} ||X - WH||_F^2 + \zeta \|W\|_F^2 \\ =& arg \min_{W \geq 0} ||(X, 0_{m \times k}) - W(H, \sqrt{\zeta} I_k)||_F^2 \\ =& arg \min_{W \geq 0} ||\widetilde{X} - W\widetilde{H}||_F^2 \end{aligned} \tag{4.10}$$

where $\widetilde{X} = (X, 0_{m \times k})$ and $\widetilde{H} = (H, \sqrt{\zeta} I_k)$.

The updating rule for W to [8]reduce the objective function can be either of the following ones, which can be proven in a similar way in [8, 18],

$$W_{ij} = W_{ij} \frac{(\widetilde{X}\widetilde{H}^T)_{ij}}{(W\widetilde{H}\widetilde{H}^T)_{ij}} \tag{4.11}$$

$$W_{ij} = W_{ij} \sqrt{\frac{(\widetilde{X}\widetilde{H}^T)_{ij}}{(W\widetilde{H}\widetilde{H}^T)_{ij}}} \tag{4.12}$$

Update H

Now we minimize the objective function with respect to H given W.

$$\begin{aligned} H &= arg \min_H O_{MM-NMF} \\ &= arg \min_H ||X - WH||_F^2 + \lambda \sum_j \|H_{.j}\|_1^2 + \eta tr(HGH^T) \\ &= arg \min_H ||\begin{pmatrix} X \\ 0_{1\times n} \end{pmatrix} - \begin{pmatrix} W \\ \sqrt{\lambda} e_{1\times k} \end{pmatrix} H||_F^2 + \eta tr(HGH^T) \\ &= arg \min_H ||\widetilde{X} - \widetilde{W}H||_F^2 + \eta tr(HGH^T) \end{aligned} \tag{4.13}$$

where $\widetilde{X} = \binom{X}{0_{1\times n}}$ and $\widetilde{W} = \binom{W}{\sqrt{\lambda}e_{1\times k}}$.

This optimization step can be done efficiently using the following updating rule, which can be proven in a similar way in [18],

$$H_{ij} = H_{ij} \sqrt{\frac{(\widetilde{W}^T\widetilde{X} + \eta HG^-)_{ij}}{(\widetilde{W}^T\widetilde{W}H + \eta HG^+)_{ij}}} \tag{4.14}$$

where

$$\begin{aligned} G &= G^+ - G^- \\ G_{ij}^+ &= \frac{|G_{ij}| + G_{ij}}{2} \\ G_{ij}^- &= \frac{|G_{ij}| - G_{ij}}{2} \end{aligned} \tag{4.15}$$

4.3.6 Convergence analysis

Since both updating methods for W and H are non-increasing, and the objective function clearly has a lower bound, for example, 0, thus the algorithm will converge.

4.4 Experiments

4.4.1 Experimental settings

We conduct clustering experiments on real data sets: the ORL face database, and the USPS handwritten digits.

The ORL face database contains ten images for each of the forty human subjects, which are taken at different times, under different lighting condition, with different facial expression and with/without glasses. All faces are resized to 32×32 for efficiency. Figure 4.1 shows some sample images from ORL data set.

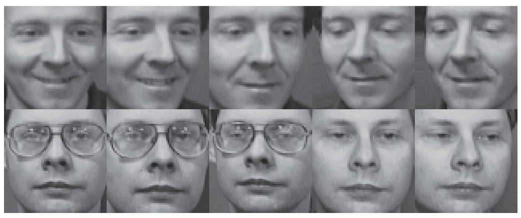


Figure 4.1.: Examples of ORL face dataset.

The USPS digit dataset contains 8-bit gray scale images of ”0” through ”9”. Each image is of size 16×16, and there are 1100 images for each class. Figure 4.2 shows some images from this dataset.

To evaluate the performance of clustering, we use two metrics [53, 54]: 1, accuracy; 2, normalized mutual information(NMI).

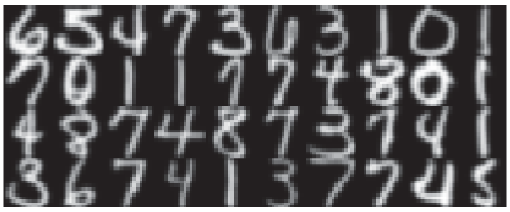


Figure 4.2.: Examples of USPS handwritten digits.

The clustering algorithm is tested on N samples. For a sample x_i, the cluster label is denoted as r_i, and ground true label is t_i. The accuracy is defined as follows:

$$accuracy = \frac{\sum_{i=1}^{N} \delta(t_i, map(r_i))}{N} \tag{4.16}$$

where $\delta(x, y)$ is equal to 1 if x is equal to y, and 0 otherwise. Function $map(x)$ is the best permutation mapping function gained by Kuhn-Munkres algorithm [55], which maps cluster to the corresponding predicted label. So, we can easily see that the more labels of samples are predicted correctly, the greater the accuracy is.

For the second metric, let C denote the cluster centers of the ground truth, and C' denote the cluster centers by clustering algorithm. The mutual information is defined as follows:

$$MI(C, C') = \sum_{c \in C; c' \in C'} p(c, c') log \frac{p(c, c')}{p(c)p(c')} \tag{4.17}$$

where $p(c)$ and $p(c')$ are the probabilities that a document belongs to cluster c and c' respectively, and $p(c, c')$ is the probability that a document jointly belongs to cluster c and cluster c'. The normalized mutual information(NMI) is defined as follows:

$$NMI(C, C') = \frac{MI(C, C')}{max((H(C)), (H(C')))} \tag{4.18}$$

where $H(C)$ and $H(C')$ are the entropies of C and C'. We can easily see that NMI measures the dependency of two distributions, and higher value of NMI means greater similarity between two distributions.

4.4.2 Clustering on human faces

We conduct ten independent experiments on the ORL face dataset. In each experiment, we randomly select ten subjects, and get 100 images since there are ten images for each subject. Then the clustering algorithms are run on the 100 images. Thus X is a matrix with the size of 1024×100. W is a matrix with size of 1024×10 and H is of size 10×100. We don't list the performance SNMF, because according to our experimental results SNMF cannot outperform NMF, which means the best choice of the parameter λ is 0. When λ is set to 0, SNMF is equivalent to NMF.

As mentioned above, ζ is simply fixed as the maximum value of the input matrix. The parameter λ is searched within $[0, 0.5]$. However, this parameter is not critical as we will see MM-NMF2, which simply sets λ to 0, still has competitive performance. The parameter η reflects how reliable the structure information encoded by S is. Generally speaking, if there are more samples, the S will be more reliable since the samples on each manifold will be denser, and the η should have larger value. In experiments, η is set by searching only on the grid of $\{0.1, 1, 10, 100, 1000\}$. In the experiments on face clustering, our λ for MM-NMF is 0.01, and η for MM-NMF and MM-NMF2 is 1.

The accuracy and NMI of different algorithms are shown in Table 4.1 and Table 4.2, respectively. The average performance of the algorithms is listed in the bottom lines of the tables.

From the results, we can see that our algorithms MM-NMF2 and MM-NMF outperform traditional NMF by about ten percent, and MM-NMF is slightly better than MM-NMF2, which means the sparseness regularizer is helpful. In other words, the sparseness introduced by nonnegativity constraint is not enough for this task, and we need extra sparseness regularization.

Table 4.1: Clustering accuracy on ORL face dataset (%).

Method	K-means	NMF	MM-NMF2	MM-NMF
1	71.0	67.0	66.0	**82.0**
2	60.0	54.0	**80.0**	79.0
3	58.0	59.0	81.0	**84.0**
4	58.0	61.0	**79.0**	78.0
5	**70.0**	52.0	68.0	64.0
6	63.0	62.0	73.0	**75.0**
7	66.0	65.0	72.0	**80.0**
8	48.0	61.0	**67.0**	64.0
9	59.0	65.0	54.0	**72.0**
10	55.0	**59.0**	**59.0**	56.0
Average	60.8	60.5	69.9	**73.4**

Table 4.2: Clustering NMI on ORL face dataset (%).

Method	K-means	NMF	MM-NMF2	MM-NMF
1	80.4	69.3	77.8	**88.8**
2	73.4	62.1	82.1	**83.6**
3	72.4	68.8	82.2	**82.6**
4	69.1	66.2	81.3	**83.5**
5	**80.5**	58.6	75.2	70.3
6	68.8	63.2	75.5	**80.6**
7	76.2	77.8	79.7	**85.2**
8	59.5	63.1	67.8	**70.1**
9	67.4	64.3	63.6	**72.8**
10	**65.4**	62.9	64.1	64.9
Average	71.3	65.6	74.9	**78.2**

According to our experiments, the performance is not sensitive to the value of η. For example, when η is set on 0.1, the average accuracy of MM-NMF is 68.6%, which is still much better than NMF.

After the matrix factorization, we gain a nonnegative basis matrix W with size of 1024×10. Ideally, each column should represent a human subject. We visualize these basis learned by NMF and MM-NMF in Figure 4.3 and Figure 4.4, respectively. From these basis, we can easily see that MM-NMF learns better representation for each class.

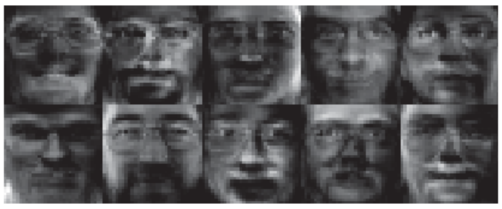


Figure 4.3.: Face basis learned by NMF.

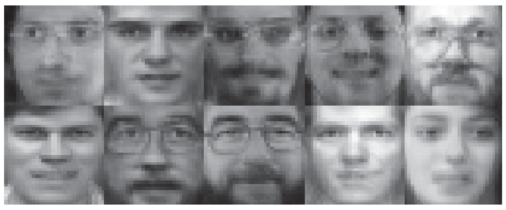


Figure 4.4.: Face basis learned by MM-NMF.

4.4.3 Clustering on handwritten digits

For experiment on USPS digits, we randomly select 1000 samples from the dataset. Thus X is a matrix with the size of 256×1000, W is of size 256×10 and H is of size 10×1000. The scheme to set parameters here is the same as face clustering. The λ here is 0.4, and η is set to 100.

Table 4.3: Clustering accuracy on USPS digit dataset (%).

Method	K-means	NMF	MM-NMF2	MM-NMF
1	64.5	54.4	**69.0**	64.3
2	56.8	48.1	**65.9**	63.6
3	56.9	55.4	**62.0**	61.2
4	50.9	52.3	55.9	**64.5**
5	58.4	53.2	61.0	**61.5**
6	65.8	52.2	**68.4**	65.9
7	63.8	59.1	53.8	**65.5**
8	**65.8**	50.5	54.6	**65.8**
9	**65.0**	47.5	56.8	60.1
10	62.6	55.7	66.3	**68.1**
Average	61.1	52.8	61.3	**64.1**

Table 4.4: Clustering NMI on USPS digits dataset (%).

Method	K-means	NMF	MM-NMF2	MM-NMF
1	59.7	51.3	**60.9**	58.8
2	61.3	46.4	**65.0**	59.6
3	58.5	49.9	58.9	**60.3**
4	54.0	46.7	53.5	**60.8**
5	**59.0**	45.8	56.0	57.4
6	63.2	50.0	**66.2**	61.6
7	59.4	55.6	53.1	**61.2**
8	63.2	50.9	59.2	**64.4**
9	**63.9**	47.3	58.5	58.5
10	59.6	53.4	62.6	**64.0**
Average	60.2	49.7	59.4	**60.9**

The accuracy and NMI of different algorithms are shown in Table 4.3 and Table 4.4, respectively. The average performance of the algorithms is listed in the bottom lines of the tables.

From the results, we can also see that our algorithms MM-NMF2 and MM-NMF outperform traditional NMF by about ten percent in both accuracy and NMI, and MM-NMF is slightly better than MM-NMF2, which again shows the sparseness regularizer is helpful. Here K-means also has comparable good performance. It outperforms traditional NMF, but it still cannot outperform our MM-NMF.

In the experiments on digits, we have more examples than experiments on faces, thus the η has larger value as mentioned above. Experiments are also conducted when η is set as 10 and 1, and MM-NMF still generates robust results with the average accuracy as 61.2% and 60.7%, which are still much better than NMF.

In experiments, each column of W should represent a digit. We visualize these basis W learned by NMF and MM-NMF in Figure 4.5 and Figure 4.6, respectively. From these two figure, we can easily see that basis learned by MM-NMF have much clearer interpretation than basis learned by NMF.

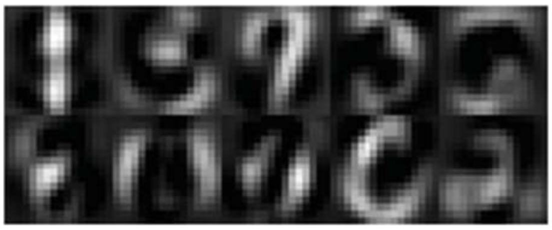


Figure 4.5.: Digit basis learned by NMF.

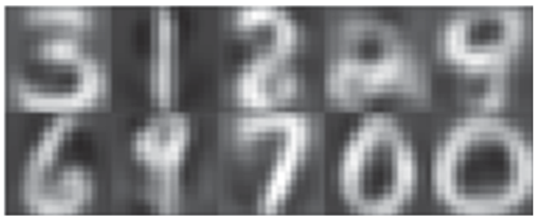


Figure 4.6.: Digit basis learned by MM-NMF.

5 CLASSIFICATION FOR DATA ON MULTIPLE MANIFOLDS

Support vector machine is one of the most popular state-of-the-art classification techniques. It is widely adopted in many real applications, such as object detection, face recognition, text categorization, etc., mainly because of its competitive practical performance and elegant theocratical interpretation. On the other hand, in many real situations, samples typically lie on low dimensional manifolds in the feature space, and thus have sparse representation over some either explicit or implicit dictionary. Specifically, it is usually believed that a sample can be represented as sparse linear combination of other samples on the same manifold. The sparsity of representation, which reflects the structure of underlying manifolds, has been deeply explored and proven to be critical for the performance of classification,and the sparse representation based technique has been a state of art approach for the task of face recognition. To benefit from the underlying low dimensional manifold structure, this chapter proposes a method named Sparsity Preserving Support Vector Machine(SP-SVM). It explicitly considers the sparse representation of samples while maximizing the margin between these from different classes, and thus inherits both the discriminative power of support vector machine and the merits of sparsity. A set of experiments on real benchmark data sets show that SP-SVM achieves significantly higher precision on face recognition than traditional SVM, sparse representation based method and classical nearest neighbor classifier.

5.1 Introduction

Pattern classification is one of the most important problems in machine learning, and support vector machine(SVM) [19, 20] is one of the most successful approaches. It is widely used in many real applications, such as face recognition [21], object detection [22], text categorization [23], and etc. In many of these applications, SVM is able to achieve very competitive performance compared with other existing approaches. SVM learns a data classifier in a fully supervised manner. Given a set of labeled data samples, it aims to find the hyperplane which separates samples with different labels with the largest margin. Later, it is extended to semi-supervised fashion [42]. This semi-supervised version of SVM tries to preserve the neighborhood relation among the unlabeled samples while searching for the separating hyperplane for those labeled samples. In this manner, both the labeled and unlabeled samples contributes the final solution. Furthermore, maximum margin clustering [43], which can be viewed as a unsupervised counterpart of SVM, tries to find hyperplane which separates the unlabeled samples with largest margin. Recently, with the rapidly increasing size of data the speed of training becomes a concern for users. To speed up the training process of SVM and thus to enable it to deal with large scale data, different optimization approaches are proposed, such as Pegasos [56], LIBLINEAR [57], and StreamSVM [58]. With the help of these techniques, SVM is able to deal with large scale data in relatively short time.

SVM is trained and tested on samples, which typically lie on lower dimensional manifolds of the feature space. A popular assumption is that a sample can be represented as a sparse linear combination of other samples on the same manifold. For classification, the samples are drawn from different classes, which are typically on different manifolds. For example, in face recognition, the face images of the same person are on the same manifold while faces of different people lie on different manifolds. A more subtle assumption is that samples within the same class may still be on a mixture of manifolds. For example, the face images of the same person with different facial expressions but same lighting condition are on a manifold, while the images with same facial expression but different lighting conditions are on another manifold. Because of these underlying low dimensional manifold structure of the data distribution, the samples that classifiers deal with usually have sparse representations with respect to some certain dictionary of base elements.

Sparse representation has been a surge of interest in many communities, such as computer vision, signal processing [52, 59, 60]. And sparsity is believed to be critical for classification of high dimensional data samples [44], such as face images.

This chapter aims to combine the discriminative power of support vector machine and the merits of sparse representation. The proposed method explicitly explores the local sparse structure in the training samples, which results from the underlying manifold structure. While seeking the separating hyperplane with large margin, the proposed algorithm tries to preserve the local sparse structure, thus we call this method Sparsity Preserving Support Vector Machine(SP-SVM).

The algorithm is then evaluated on multiple real world data sets. And, it has been shown that the proposed algorithm achieves significantly better performance in terms of precision than traditional SVM, which we believe benefits from exploring the sparsity in the representation of samples.

The rest of this chapter is organized as follows. First it reviews the concepts of sparse representation and SVM. Then the proposed algorithm SP-SVM is presented. Furthermore, it describes and analyzes the experimental results on real world data sets.

5.2 Review of Related Techniques

In this section, we briefly review the large margin classification algorithm SVM and the concept of sparse presentation.

5.2.1 Support vector machine

Given a set of labeled samples $\{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^m$ is the feature vector of i_{th} sample and y_i is the corresponding label(for binary classification, $y_i \in \{-1, +1\}$), SVM learns a hyperplane that maximizes the margin between samples with different labels.

In order to learn the hyperplane $w \in \mathbb{R}^m$, SVM solves the following optimization problem:

$$\min_w C \sum_i \max(0, 1 - y_i w^T x_i) + \frac{1}{2}\|w\|^2. \tag{5.1}$$

The first term in (5.1) is the hinge loss, the second term is a regularizer for w, and C is a parameter controlling the tradeoff between the hinge loss and the regularization on w. The explicit bias term is omitted here to simplify the problem while keeping it entirely general since the bias could be easily introduced as an additional coordinate in the data.

5.2.2 Sparse representation

An observed sample $x \in R^m$ has a sparse representation with respect to a dictionary $D \in \mathbb{R}^{m \times p}$. This means:

$$x = D\beta, \tag{5.2}$$

where the coefficient vector $\beta \in \mathbb{R}^p$ is sparse.

In real situations, usually the observation x is polluted by noise, thus we have the following equation:

$$x = D\beta + \epsilon, \tag{5.3}$$

where $\epsilon \in \mathbb{R}^m$ denotes the noise term.

It is more interesting that given the dictionary D and the sample x, how to estimate the sparse representation β? Lasso [61] is a popular approach for this task since $L1$ norm penalty encourages the sparsity of the representation [59] and its minimization can be efficiently solved:

$$\hat{\beta} = \arg\min_{\beta} \frac{1}{2}\|x - D\beta\|_2^2 + \lambda\|\beta\|_1. \tag{5.4}$$

Lasso is simple and effective, and it is known to be able to select the few important base elements from the dictionary D for the given sample x.

5.3 Our Approach

Here we describe the formulation of the proposed approach, which explicitly considers the sparse representation of samples on a mixture of manifolds while learning a large margin classifier.

5.3.1 Formulation

Given a set of training samples, SVM learns a separating hyperplane by maximizing the margin between samples with different labels. Even if a sample is dense in the original feature space, it may still have sparse representation with respect to some dictionary of base elements. One possible explanation is that the samples reside on different manifolds, and a sample has a sparse representation on its manifold [16, 28], such as face images. Face images of the same subject reside on the same manifold while images of different subjects reside on different manifolds. However, SVM ignores the implicit sparse representation of the samples, and thus ignores the structure of the underlying manifold as a result.

SVM are trained on samples from different classes, thus it is safe to assume these training samples are from different manifolds. To encode the multiple manifold structure, the implicit sparse representation of each sample with respect to the other samples in the same class is calculated. In this way, a sparse graph connecting n nodes is created to encode the manifold structure(n is the number of samples, and each node corresponds to a sample). Note, we do not simply connect neighbors according to the Euclidean distance, because even the samples in the same class may still be from different manifolds(e.g. the face images of a person with different facial expressions but same lighting condition are on a manifold, while the images with same facial expression but different lighting conditions are on another manifold), and samples close to each other may still be on different manifolds, especially near the intersection of manifolds. Sparse representation not only weighs more on the selected samples that are close to it, but also tends to select the samples on the same manifold.

The graph created in the way described above results from the sparse representation of each sample with respect to other samples, and thus conveys information of the underlying manifold structure. Our proposed classification algorithm incorporates an preservation term of this sparse graph into the conventional SVM objective function, and thus the proposed algorithm is able to preserve the graph structure in the output space while learning a large margin classifier.

5.3.2 Sparse representation for structure encoding

Let x_i be an arbitrary training sample and y_i be the corresponding label. x_i can be represented as a sparse linear combination of the other training samples in the same class. Let X be the matrix composed of all the training samples, i.e. $X = [x_1, x_2, ..., x_n]$, each column of this matrix is the feature vector of a training sample.

To estimate the sparse structure of x_i, we solve the following optimization problem:

$$\begin{aligned} S_{.i} = &\arg\min_{\beta} \frac{1}{2}\|x_i - X\beta\|_2^2 + \lambda\|\beta\|_1 \\ s.t.\ \ &\beta_i = 0 \\ &\beta_j = 0,\ if\ y_j \neq y_i. \end{aligned} \tag{5.5}$$

Sometimes, we are more interested in the local sparse structure. In this case, x_i is represented as a sparse linear combination of its neighbors. Let $\mathcal{N}(x_i)$ denote the set of neighboring samples of x_i, either k nearest neighbors or ϵ neighbors. The related optimization problem then is modified into:

$$\begin{aligned} S_{.i} = &\arg\min_{\beta} \frac{1}{2}\|x_i - X\beta\|_2^2 + \lambda\|\beta\|_1 \\ s.t.\ \ &\beta_i = 0 \\ &\beta_j = 0,\ if\ y_j \neq y_i\ or\ x_j \notin \mathcal{N}(x_i). \end{aligned} \tag{5.6}$$

The neighboring constraint introduces locality of the sparse representation, and it also highly reduces computational cost especially when the number of training samples is large.

Let $S \in \mathbb{R}^{n \times n}$ be the matrix composed of the sparse representation of all the training samples, i.e. $S = [S_{.1}, S_{.2}, ..., S_{.n}]$. It describes the geometrical information conveyed by the training data.

5.3.3 Sparsity preserving support vector machine

For a sample x_i with sparse representation $S_{.i}$, its identity is closely related to the samples corresponding to nonzero coefficients of $S_{.i}$.

Since the geometrical structure information encoded by the sparse representation S is critical for classification [44], we would like to preserve this while training the discriminative large margin classifier. For convenience, we introduce two new notations. Let $Y \in \mathbb{R}^{1 \times n}$ be a row vector composed of the labels of all the training samples, i.e. $Y = [y_1, y_2, ..., y_n]$. Let $f(X) \in \mathbb{R}^{1 \times n}$ be the output of the classifier when applied to all the training samples, i.e. $f(X) = [f(x_1), f(x_2), ..., f(x_n)]$. Specifically, for linear SVM, $f(X) = w^T X$. Again, the bias term is omitted here for convenience without sacrificing the generality of the problem.

In the input space, we have $X \approx XS$ because of the first terms in the optimization problems (5.5) and (5.6); to preserve the structure information, it is preferable to ensure $f(X) \approx f(X)S$, which can be gained by minimizing:

$$\begin{aligned}
&\sum_{i=1}^{N} \|f(x_i) - f(X)S_{.i}\|_2^2 \\
=&\|f(X) - f(X)S\|_2^2 \\
=&\|f(X)(I - S)\|_2^2 \\
=&f(X)(I - S)(I - S)^T f(X)^T \\
=&w^T X(I - S)(I - S)^T X^T w,
\end{aligned} \tag{5.7}$$

where $S_{.i}$ is the i_{th} column of the matrix S.

Incorporating the term above to SVM, we get the following optimization problem:

$$\begin{aligned}\min_w C \sum_i \max(0, 1 - y_i w^T x_i) + \frac{1}{2}\|w\|^2 \\ + \frac{\nu}{2} w^T X(I-S)(I-S)^T X^T w\end{aligned} \tag{5.8}$$

The first two terms are inherited from the conventional SVM: the first term is the hinge loss of training samples, and the second term is standard $L2$ norm regularizer. The third term tries to preserve the sparse structure information, and the parameter ν controls how much weight we put on the structure preservation. We call this technique Sparsity Preserving Support Vector Machine(SP-SVM) since it tries to preserve the structure reflected by sparse representation while learning a large margin classifier.

When ν is equal to 0, SP-SVM reduces to the conventional SVM; when C is equal to 0, it reduces to learning a mapping direction which preserves the sparse structure rather than a classifier since it does not incur any loss if a sample is wrongly classified. More interestingly, when the parameter λ in Equation 5.6 is equal to 0, then minimizing the objective function above turns out to be learning a large margin classifier that preserves the neighborhood information.

The objective function of SP-SVM is in the form of quadratic programming, which is the same as conventional SVM. This can be efficiently solved by standard optimization toolboxes. Also, many of the techniques [56, 62] that help to speed up the conventional SVM could be easily introduced to SP-SVM without much modification.

Overall, the training process of SP-SVM involves the computation of the sparse graph S, and a quadratic programming problem. The flow is listed in the Algorithm 4. The testing process of SP-SVM is exactly the same as conventional SVM. The label of a testing sample x is predicted as $sign(w^\top x)$, where $sign(.)$ is the simple sign function.

Table 5.1: Performance comparison on three benchmark datasets.

Methods	NN	SR	SVM	SP-SVM
Extended Yale B	51.45 ± 0.75	95.44 ± 0.57	94.83 ± 0.65	$\mathbf{98.13 \pm 0.46}$
CMU PIE	43.66 ± 0.88	84.55 ± 1.06	85.10 ± 0.81	$\mathbf{90.02 \pm 0.54}$
AR	49.72 ± 1.67	92.73 ± 0.33	93.67 ± 0.76	$\mathbf{97.60 \pm 0.36}$

5.4 Experiments

In this section, we describe the experimental settings, and show and briefly analyze the results.

5.4.1 Experimental settings

In the experiments, three benchmark data sets including Extended Yale B database [63], AR database [64], and CMU PIE database [65] are utilized to evaluate the performance of our proposed algorithm. The images are cropped to 32×32 and power normalized [66]. For each data set, the data are randomly split into training set and testing set. We randomly select images per category for training and the rest for testing. Since for the face recognition task we usually have very limited training samples for each person in real situation, our experiments use 10 training samples per category. It is shown that for the Extended Yale B database and AR database, experiments with 10 training samples per category already achieve very high precision(98.13% and 97.60%); for CMU PIE database, the precision is

Algorithm 4 SP-SVM Training

```
Require: Data samples $X = [x_1, x_2, ..., x_n]$, sample labels $Y = [y_1, y_2, ..., y_n]$, $\lambda$, $\nu$ and $k$
1: for $i = 1$;$k \leq n$;$i$++ do
2:     Compute $S_i$ according to Equation 5.6
3: end for
4: Formulate $S = [S_1, S_2, ..., S_n]$
5: Compute the $w$ by solving the quadratic programming problem in Equation 5.8
6: return $w$
```

around 90%, which allows space to improve. Then, specifically, for CMU PIE database, we conduct more experiments with more training samples per category, and the details are reported later.

To make the results more convincing, the experimental process is repeated 10 times, and the mean and standard deviation of the classification accuracy are recorded.

Sparse representation (SR) [44] explores the sparsity of the given sample, achieves high precision in recognition, and is considered as one of the state of art approaches for face recognition. SVM is a popular and competent technique for many classification problems including face recognition. In our experiments, the proposed SP-SVM algorithm is compared with these two algorithms, as well as the classical and popular approach: nearest neighbor classification(NN).

Given a testing sample, SR calculates its sparse linear representation with respect to all the training sample by Equation 5.4, where x is the testing sample and D is composed of all the training samples. Feature-sign search algorithm [67] is used for solving sparse representation. Then the identity of the testing samples is determined by the fitting error. For SVM, the implementation of LIBSVM [68] is adopted for its robustness and popularity. For SP-SVM, S is calculated according to the Equation 5.6, where $\mathcal{N}(x_i)$ denotes the set of k nearest neighbors of x_i. One against all multi-classification strategy is adopted by both SVM and SP-SVM.

In our experiments, there are four parameters to set: C, ν, k and λ. C is varying on the grid of $\{2^0, 2^1, \cdots, 2^5\}$ for both SVM and SP-SVM. ν is varying on the grid of $\{2^{-1}, 2^0, \cdots, 2^4\}$. λ is studied on the grid of $\{10^{-4}, 10^{-3}, \cdots, 10^1\}$ for sparse representation algorithm. For k, we consider the set $\{1, 3, 5, 7, 9\}$. For the main experiments, k is simply equal to 5, since the performance is not sensitive to k when $k \geq 3$ as we will show below.

Figure 5.1.: Examples of the Extended Yale B dataset.

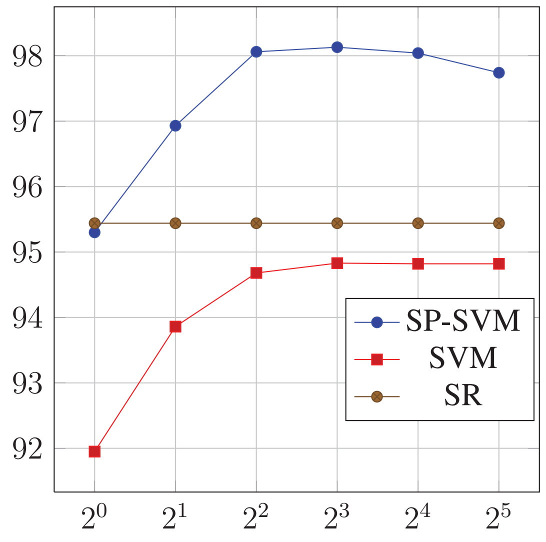


Figure 5.2.: Precision(%) of SP-SVM, SVM, and SR with varying parameter C on Extended Yale B database ($\lambda = 0.01$, $\nu = 2^2$, $k = 5$ for SP-SVM, $\lambda = 0.01$ for SR)

5.4.2 Experimental results on extended Yale B database

The Extended Yale B database contains 2414 frontal face images of 38 individuals, which were captured under varying illumination conditions. Figure 5.1 shows some sample images from this database.

Figure 5.2 shows the precision of SP-SVM, SVM, and SR with varying parameter C. As C increase, the performance of SVM and SP-SVM $\nu = 2^2$ firstly increases, till to the optimal value around $C = 2^3$, and then it decreases. From the figure, we can see that SR outperforms SVM, which does not consider the sparse representation. However, when the large margin classifier is coupled with sparsity preserving, the resulting algorithm, SP-SVM, achieves a precision of 98.13%, which is far beyond SR(95.44%). This demonstrates the criticality of the sparsity preserving. SR method considers the compact representation of the data samples and explores the underlying structure, while SVM focuses on the margin between different classes. Both of these two facets are important for the classification, and thus when combining these two, the precision is significantly boosted.

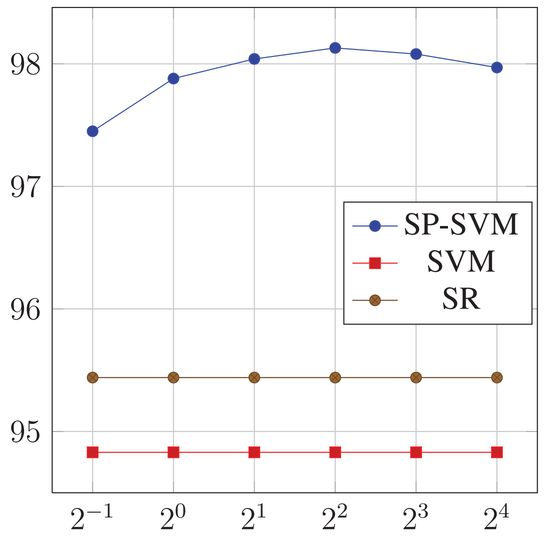


Figure 5.3.: Precision(%) of SP-SVM with varying parameter ν on Extended Yale B database ($\lambda = 0.01$, $C = 2^3$, and $k = 5$ for SP-SVM, $C = 2^3$ for SVM, $\lambda = 0.01$ for SR)

The effect of parameter ν is also studied. We fix $C = 2^3$, and vary the value of ν. The precision($\%$) of SP-SVM is reported in Figure 5.3. As ν increase, the performance of SP-SVM increases, till to the optimal $\nu = 2^2$. We can also see that regardless of what value the ν is(as long as ν is in that range), the precision of SP-SVM always outperforms both SR and SVM. This further demonstrates the importance of the combining of these two facets. Actually, it is not difficult to see that when $\nu = 0$, SP-SVM degenerates into conventional

SVM; when $\nu = +\infty$, the learning process of SP-SVM becomes the same as SR, though the prediction is different.

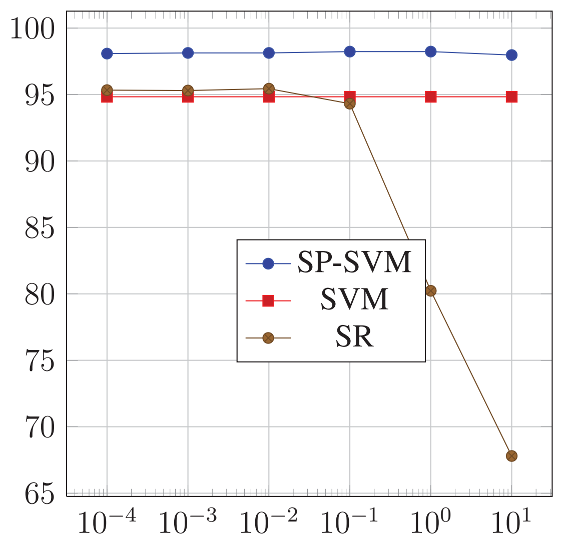


Figure 5.4.: Precision(%) of SP-SVM and SR with varying parameter λ on Extended Yale B database ($C = 2^3$, $\nu = 2^2$, and $k = 5$ for SP-SVM, $C = 2^3$ for SVM)

The effect of λ is shown in Figure 5.4, which shows the precision(%) of SP-SVM and SR with varying parameter λ. The precision achieved by SR algorithm is very sensitive to the value of λ, especially when $\lambda > 0.1$. When $\lambda = 10$, the precision is only 67.80%. However, the precision achieved by SP-SVM changes slowly with varying λ. Again, this insensitivity to λ should be due to the combination of the large margin and sparse representation. The effect of parameter λ for the other two data sets is nearly the same as for Extended Yale B database, i.e., the performance is not sensitive to λ, and thus we omit these figures.

Figure 5.5 shows the precision(%) of SP-SVM with varying parameter k. The experiments are conducted for $k \in \{1, 3, 5, 7, 9\}$. When $k = 1$, the precision is very low. When $k \geq 3$, the precision is relatively consistent. This probably is because it is difficult to fit the target sample with only one neighbor. When there are more samples, sparse representation will always be able to select the homogenous neighbors to fit the data point.

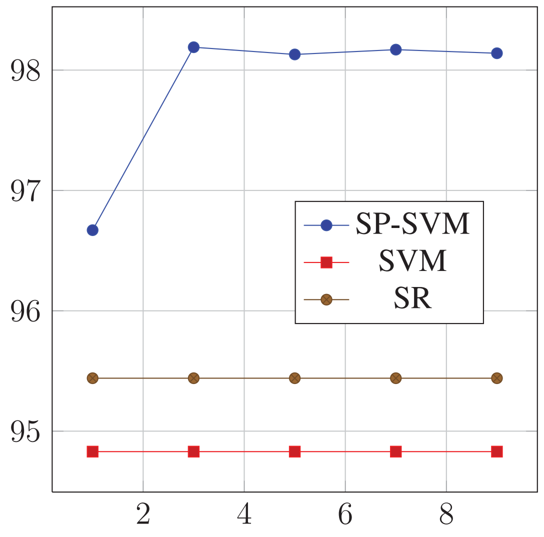


Figure 5.5.: Precision(%) of SP-SVM with varying parameter k on Extended Yale B database ($C = 2^3$, $\nu = 2^2$, and $\lambda = 0.01$ for SP-SVM, $C = 2^3$ for SVM, $\lambda = 0.01$ for SR)

After all, the experimental results (when $\lambda = 0.01$, $C = 2^3$, $\nu = 2^2$ and $k = 5$) are listed in Table 5.1, which shows the precision of SP-SVM algorithm is 3.30% higher than SVM and 2.69% higher than SR, while the standard deviation of all the three algorithms is smaller than 0.7%.

5.4.3 Experimental results on CMU PIE database

In the CMU PIE database, there are totally $41,368$ images of 68 people, each person under 13 different poses, 43 different illumination conditions, and with 4 different expressions. Thus, the images in the same category may still lie on multiple manifolds. We select the images with five near frontal poses (C05, C07, C09, C27, C29) and all different illuminations and expressions. There are about 170 images for each individual and totally $11,554$ images. Figure 5.6 shows some sample images from this database.

The results of SR, SVM, NN, and SP-SVM on PIE database are listed in Table 5.1 when $\lambda = 0.01$, $C = 2^3$, $\nu = 2^1$ and $k = 5$. The precision of SP-SVM is 4.92% higher than SVM

Figure 5.6.: Examples of CMU PIE dataset.

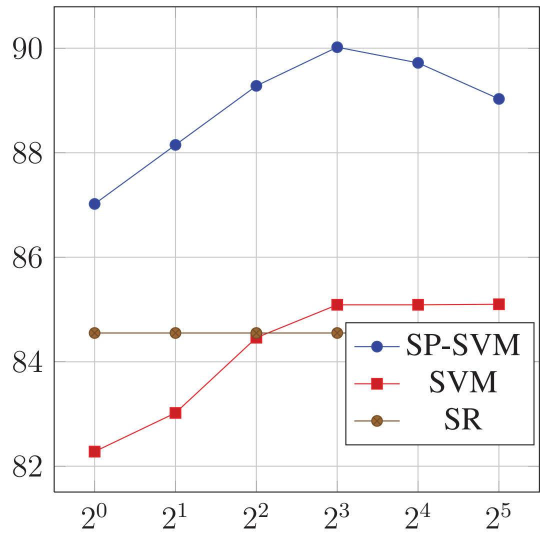


Figure 5.7.: Precision(%) of SP-SVM, SVM, and SR with varying parameter C on CMU PIE database ($\lambda = 0.01$, $\nu = 2^1$, $k = 5$ for SP-SVM, $\lambda = 0.01$ for SR)

and 5.47% higher than SR. Both of these gains are significant since the standard deviation is quite small. Moreover, Figure 5.7 shows the precision of SP-SVM, SVM and SR with varying parameter C, and Figure 5.8 studies the effect of varying ν. Figure 5.9 again shows the insensitivity of the performance with respect to a varying when $k \geq 3$.

We find that for this dataset, the precision produced by 10 training images per category is only about 90%. Then, a further experiment is conducted with 20 training images per

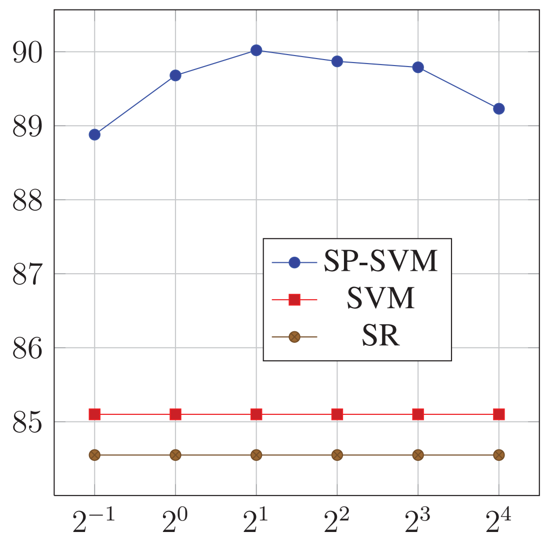


Figure 5.8.: Precision(%) of SP-SVM with varying parameter ν on CMU PIE database ($\lambda = 0.01$, $C = 2^3$, and $k = 5$ for SP-SVM, $C = 2^5$ for SVM, $\lambda = 0.01$ for SR)

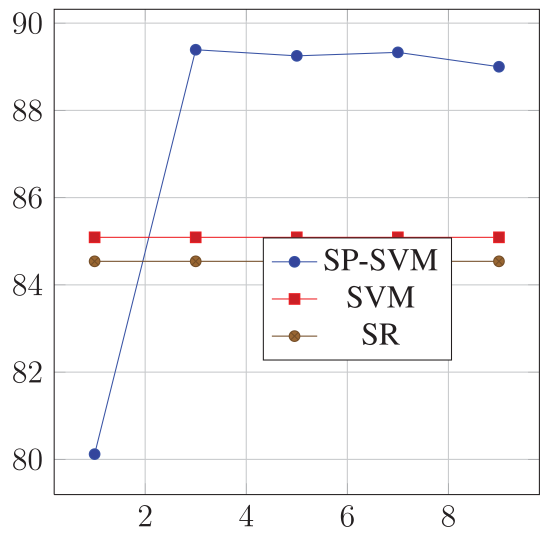


Figure 5.9.: Precision(%) of SP-SVM with varying parameter k on CMU PIE database ($C = 2^2$, $\nu = 2^1$, and $\lambda = 0.05$ for SP-SVM, $C = 2^4$ for SVM, $\lambda = 0.01$ for SR)

category. And with this new setting, SR achieves an average precision of 92.65% with a standard deviation of 0.53%, conventional SVM achieves 93.10% with a standard deviation of 0.42%, and the proposed SP-SVM still gets the highest performance, which is 94.88%

with a standard deviation of 0.22%. Although the precision of all algorithms increase as the number of training samples increase, the proposed SP-SVM still outperforms the other two baseline algorithms significantly.

5.4.4 Experimental results on AR database

Figure 5.10.: Examples of AR Database

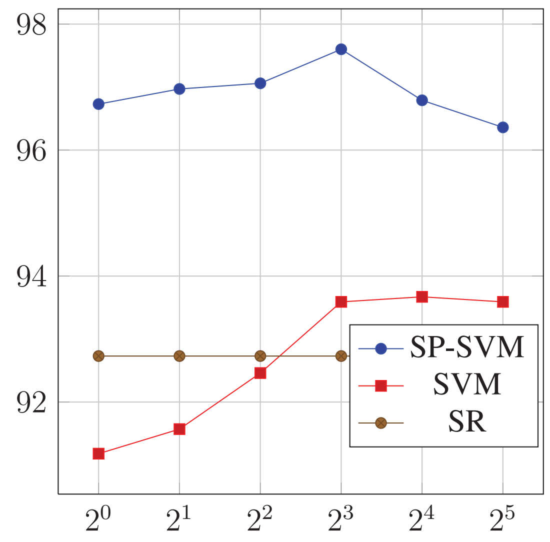


Figure 5.11.: Precision(%) of SP-SVM, SVM, and SR with varying parameter C on AR database ($\lambda = 0.01$, $\nu = 2^2$, $k = 5$ for SP-SVM, $\lambda = 0.01$ for SR)

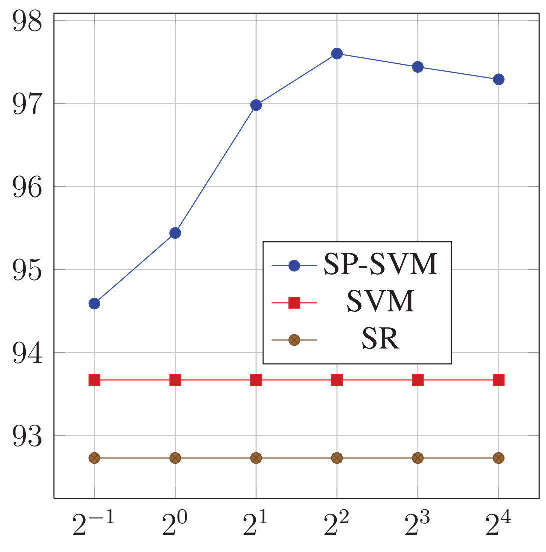


Figure 5.12.: Precision(%) of SP-SVM with varying parameter ν on AR database ($\lambda = 0.01$, $C = 2^3$, and $k = 5$ for SP-SVM, $C = 2^4$ for SVM, $\lambda = 0.01$ for SR)

There are over $4,000$ frontal images for 126 individuals in AR database. We choose a subset consisting of 50 male and 50 female categories. There are 26 images for each category. Compared with two other data sets, the AR database includes more facial variations, such as illumination change, various expressions, and facial disguises. Please refer to Figure 5.10 for examples.

Again, the face recognition experiments are conducted with the standard setting. The results with $\lambda = 0.01$, $C = 2^3$, $\nu = 2^2$ and $k = 5$ are reported in Table 5.1. SP-SVM algorithm has much higher precision than SVM(by 3.93%) and SR (by 4.87%). Also, the standard deviation is quite small so that we can conclude the SP-SVM outperforms the others significantly. Figure 5.11 shows the precision of SP-SVM, SVM and SR with varying parameter C, and Figure 5.12 shows the precision(%) of SP-SVM with varying parameter ν with $C = 2^3$. In this wide range of C, SP-SVM consistently outperforms conventional SVM and SR.

6 TAPERING KERNEL MATRICES TO SPARSIFY DATA RELATION

Kernel methods have been regarded as an effective approach in image processing since k-ernel is able to model nonlinear relation among images. However, when calculating the similarity induced by kernels, existing kernel methods usually incorporate irrelevant features (e.g., the background features of an object in a image), which are then inherited to kernel learning methods and thus lead to suboptimal performance. To attack this problem, we introduce a framework of kernel tapering, which is a simple and effective approach to reduce the effects of irrelevant features while keeping the positive semi-definiteness of k-ernel matrices. In theory, it can be demonstrated that the tapered kernels asymptotically approximate the original kernel functions. In practical image applications where noises or irrelevant features are widely observed, we have further shown that the introduced kernel tapering framework can greatly enhance the performance of their original kernel partners for kernel k-means and kernel nonnegative matrix factorization.

6.1 Introduction

Kernels [1–3] have been widely applied in image processing tasks, such as image clustering/classification, face recognition, object tracking, etc. They are incorporated into a number of machine learning algorithms [2–7], such as matrix factorization, k-means clustering, and support vector machine, to form new powerful methods, i.e., the so called kernel methods. In kernel methods, low dimensional input features are projected to a high dimensional space via a kernel-induced nonlinear mapping, such that more useful features hidden in the original data can be utilized. Despite their success, however, traditional kernel methods usually employ all features and instances to produce a kernel matrix with a high chance of incorporating noises and irrelevant features. For example, in object categorization, the background of an image may greatly impair the clustering accuracy. How to reduce or re-

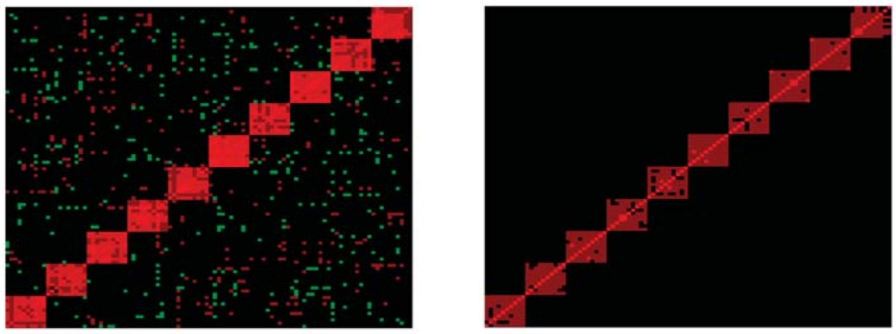


Figure 6.1.: A toy example showing the effects of kernel tapering. Left: linear kernel matrix for noisy samples; right: tapered linear kernel matrix for noisy samples.

move the effects of the background or irrelevant features in kernel construction remains an open problem.

An effective approach to attack this problem is to introducing sparsity into kernels. Sparsity is believed to be beneficial to many task [28, 41, 44, 45, 69]. Therefore, we present the idea of kernel tapering into the kernel design. Kernel tapering, also known as covariance tapering, originates from the study of Geostatistics [70, 71]. The idea is to push the distances between pairs of distant instances to infinity, thus leading to a sparse kernel matrix while keeping the positive semidefiniteness. In this way, the effects of noises when calculating the distances can be reduced to a certain level. In addition, a sparse kernel matrix is also of the demand of data analysis. For example, in the categorization of car and goat, the similarity between an image of car and an image of goat should be zero ideally. What's more, operations on sparse matrices take less computer memories and run faster. Kernel tapering removes or reduces the effects of noises while assuring that most information are kept. A toy example illustrating the effect of tapering is shown in Figure 6.1 for simulated data of ten categories with a Gaussian noise (see more details in the experimental part). It clearly shows that kernel tapering can significantly remove the noises in the kernel matrix.

To take advantage of the useful properties of kernel tapering, we apply this idea to improve kernel k-means and kernel nonnegative matrix factorization. Experimental results on benchmark data sets show that the proposed tapered kernel methods greatly improved the clustering performance.

6.2 Kernel Tapering

In this section, we begin with the review of kernel functions, then introduce the concept of tapering, and finally incorporate tapering into two machine learning algorithms: kernel k-means and kernel nonnegative matrix factorization.

6.2.1 Brief review of kernels

Before introducing the idea of kernel tapering, we give the notations as follows. Let ϕ be a mapping from the original input space $\mathcal{F}$ to the high or even infinite dimensional Hilbert space $\mathcal{H}$, i.e., $\phi : x \in \mathcal{F} \rightarrow \phi(x) \in \mathcal{H}$. Let $\kappa(x, y)$ denote the inner product defined in $\mathcal{H}$ for a certain pairs of data $x, y \in \mathcal{F}$. We also refer $\kappa(\cdot, \cdot))$ as a kernel function. A matrix $K \in R^{n\times n}$ is called a kernel matrix (also known as Gram matrix) for kernel function κ if $K_{ij} = \kappa(x_i, x_j)$ for $x_1, \ldots, x_n \in \mathcal{F}$. Theoretically, a kernel matrix K must be positive semi-definite (PSD) [72]. Typical valid kernel functions include linear kernel, RBF kernel, exponential kernel, etc.

6.2.2 Tapering

However, during the calculation of kernel matrices, the noise in features can be incorporated. To alleviate this effect, a central idea is to sparsify the kernel matrices—-to deliberately push the similarity measures between pair of distant samples to zeros. How the zeros are introduced is crucial, since the positive semi-definiteness of the kernel matrix must be kept. To this end, we introduce the concept of kernel tapering. The idea is to sparsify the kernel matrix by pushing the small values to zeros while maintaining the validity of kernel matrix(positive semi-definiteness). In Geostatistics, it has been used to reduce the computation in parameter estimation for kernel(a.k.a. covariance function) in

Gaussian process [70]. In our situation, we introduce the tapering technique not for the purely computational reasons, but with the expectation that it will result in better kernels and help boost the performance of kernel based algorithms.

A tapered kernel is to define a new kernel function such that

$$\tilde{\kappa}(x_i, x_j) = \kappa(x_i, x_j)\kappa_{taper}(g(x_i, x_j); \theta), \tag{6.1}$$

where $g(\cdot, \cdot)$ is a distance function and θ is a parameter. Note, the tapering is not restricted to specific types of kernel, instead it can be applied to any existing valid kernel. Thus, it not only keeps the validity of kernel while sparsifying it, but also preserves the richness and generality of the current family of various kernels. Then the final tapered covariance function is

$$\hat{K}(\theta) = K \circ K(\theta)$$

where $K_{ij}(\theta) = \kappa_{taper}(g(x_i, x_j); \theta)$ and $\circ$ denotes the element wise matrix product, also known as Hadamard product or Schur product. Note that the Hadamard product of two kernel matrices is also a valid kernel matrix.

To see the asymptotic properties of tapered kernels, we use a regression task as an example. In order to show the asymptotic equivalence of the mean squared prediction error for two kernel functions, we first describe the tail behaviors of the corresponding spectral densities and introducing the tail condition.

Tail Condition. Two spectral densities f_0 and f_1 satisfy the tail condition if and only if

$$\lim_{\eta \to \inf} \frac{f_1(\eta)}{f_0(\eta)} = \gamma, \ 0 < \gamma < \infty.$$

Let us consider the exponential kernel as an example. Let f_κ and f_{taper} denote the spectral densities for the exponential kernel and taper functions. Then the spectral density $f_{\hat{\kappa}}$ of the tapered covariance function $\hat{\kappa}$ is given by:

$$f_{\hat{\kappa}}(\|u\|) = \int_{\mathbb{R}^d} f_\kappa(\|u - v\|) f_{taper}(\|v\|) dv, \tag{6.2}$$

recalling that the convolution or multiplication of two functions is equivalent to, multiplication or convolution of their Fourier transforms, respectively.

Proposition 1 *Let f_{taper} be the spectral density of the taper kernel function, and for some $\epsilon \geq 0$ and $M < \inf$, satisfying*

$$0 < f_{taper}(\eta) < \frac{M}{(\sqrt{1+\gamma^2})^{1+d+\epsilon}},$$

then $f_{\hat{\kappa}}$ and f_{κ} satisfy the tail condition.

Proposition 1 is a special case of Proposition 2.2 in [71]. Then we have the following theorem

Theorem 1 *For an exponential kernel function κ satisfying the conditions in Proposition 1, then*

$$\lim_{n \to \inf} \frac{MSE(x, K \circ K(\theta))}{MSE(x, K)} = 1, \tag{6.3}$$

where the $MSE(x, K)$ denotes the mean square error of the predictor with the kernel function K on a test example x.

Theorem 1 follows from Theorem 1 in [71]. It shows that the same convergence rate as the optimal predictor using the original kernel function K.

Some examples of tapering functions [71] include:

- Spherical: $\kappa_{taper}(g;\theta) = (1 - \frac{1}{\theta}g)_+^2(1 + \frac{1}{2\theta}g)$
- Wendland$_1$: $\kappa_{taper}(g;\theta) = (1 - \frac{1}{\theta}g)_+^4(1 + \frac{4}{\theta}g)$
- Wendland$_2$: $\kappa_{taper}(g;\theta) = (1 - \frac{1}{\theta}g)_+^6(1 + \frac{6}{\theta}g + \frac{35}{3\theta^2}g^2)$

Here $x_+ = max(0, x)$.

In Figure 6.2, we visualize the shapes of different kernel functions that are purely based on distances between samples, and corresponding tapered kernels. For simplicity, we only choose kernel functions that are purely based on distances.

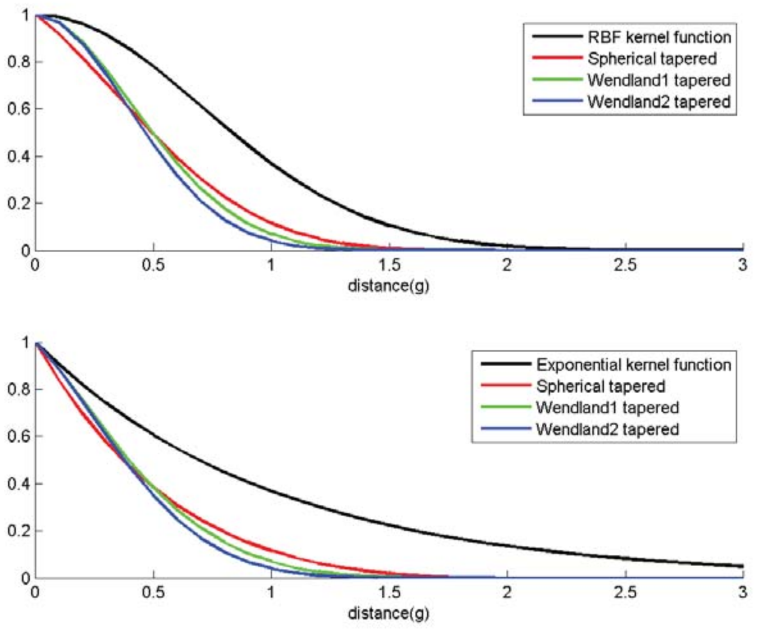


Figure 6.2.: An example showing the change of values with respect to distances for kernel functions and their tapered versions. Top: RBF kernel function and tapered RBF kernel functions; bottom: exponential kernel function and tapered exponential kernel functions.

From the figures, we can see that the new tapered kernel functions are more sparse than usual kernel functions. When the distance is large, tapered kernel function will have small or even zero value.

6.2.3 Tapered kernel methods

The kernel based machine learning methods can be easily extended once the kernel is tapered. The induced algorithms use the tapered kernels rather than the regular kernels. During the learning process, the objective function should be modified accordingly.

Here we incorporate the tapering into two kernel based machine learning techniques: kernel k-means clustering and kernel Nonnegative Matrix Factorization (kernel NMF).

Tapered kernel k-means clustering

k-means is a popular clustering method for data analysis. It aims to partition the data samples $\mathcal{S} = \{x_i \in \mathcal{F} | i = 1, 2, \ldots, n\}$ into k sets, $\{S_1, S_2, \ldots, S_k\}$, in which each sample is assigned to the set with the nearest mean by minimizing the following objective function:

$$\underset{\{S_1,S_2,\ldots,S_k\}}{\arg\min} \sum_{i=1}^{k} \sum_{x_j \in S_i} \|x_j - \mu_i\|^2, \tag{6.4}$$

where μ_i is the mean of points in S_i.

Let X be a matrix composed of all the samples, i.e. $X = [x_1, x_2, \ldots, x_n]$, and A be a clustering membership matrix, i.e. $A \in \{0,1\}^{n \times k}$, $\sum_{j=1}^{k} A_{ij} = 1$ and $n_j = \sum_{i=1}^{n} A_{ij}$. Let $\tilde{A} = A[diag(\sqrt{n_1}, \sqrt{n_2}, \ldots, \sqrt{n_k})]^{-1}$. The optimization above is equivalent to following problem [73, 74]:

$$\underset{A}{\arg\min}\ \text{trace}(X^\top X) - \text{trace}(\tilde{A}^\top X^\top X \tilde{A}). \tag{6.5}$$

With the kernel trick, kernel k-means minimizes the following objective function:

$$\underset{\{S_1,S_2,\ldots,S_k\}}{\arg\min} \sum_{i=1}^{k} \sum_{x_j \in S_i} \|\phi(x_j) - \mu_i\|_{\mathcal{H}}^2, \tag{6.6}$$

where μ_i is the mean of $\phi(x_j)$ in S_i and $\|.\|_{\mathcal{H}}$ denotes the functional norm in the Hilbert space endowed by the kernel function. And the equivalent formulation for kernel k-means is:

$$\underset{A}{\arg\min}\ \text{trace}(K) - trace(\tilde{A}^\top K \tilde{A}). \tag{6.7}$$

Once we have the kernel matrix $K \in R^{n \times n}$, the membership matrix A can be inferred by the optimizing the objective function above.

Tapered kernel k-means infers the membership matrix A by minimizing the following objective function

$$\text{trace}(K \circ K_{taper}) - \text{trace}(\tilde{A}^\top (K \circ K_{taper}) \tilde{A}). \tag{6.8}$$

Tapered kernel nonnegative matrix factorization

Nonnegative Matrix factorization (NMF) [8] is a popular technique in image processing, computer vision, data mining, etc. Given a nonnegative input data matrix X, NMF seeks two nonnegative matrices to approximate X by minimizing the following objective function:

$$(W, H) \quad = \quad \arg\min_{W,H\geq 0} \ \|X - WH\|_F^2. \tag{6.9}$$

Zhang *et al.* [4] proposed to do nonnegative matrix factorization on kernels (kernel NMF). Let $X = [x_1, x_2, ..., x_n]$ be the input nonnegative data matrix. Then, the mapped data matrix is $\phi(X) = [\phi(x_1), \phi(x_2), ..., \phi(x_n)]$. By matrix factorization technique, kernel NMF searches W_ϕ and H, such that $\phi(X) \approx W_\phi H$, where W_ϕ is the base matrix in the space $\mathcal{H}$ and H is the coefficient matrix.

The kernel matrix $K = \phi(X)^\top \phi(X) \in R^{n\times n}$, so

$$K = \phi(X)^\top \phi(X) \approx \phi(X)^\top W_\phi H. \tag{6.10}$$

Let $Y \doteq \phi(X)^\top W_\phi$, then we have $K \approx YH$. Given the input data matrix X, kernel matrix K can be computed by $K_{ij} = \langle \phi(x_i), \phi(x_j)\rangle = k(x_i, x_j)$. Then Y and H can be learned by factorizing the kernel matrix K.

Kernel NMF aims to approximate $K = \phi(X)^\top \phi(X) \in R^{n\times n}$, while Tapered kernel NMF focuses on $K \circ K_{taper} = \phi(X)^\top \phi(X) \in R^{n\times n}$. Y and H are learned by factorizing the tapered kernel matrix $K \circ K_{taper}$, where $Y \doteq \phi(X)^\top W_\phi$.

6.3 Experiments

Before evaluating the performance of tapered learning algorithms, we first design a synthetic data set to illustrate the effects of tapered kernels. We design the training data, a 100×100 matrix, to be 10 groups with each having 10 columns; each sample from the $k_{th}(k = 1, 2, ..., 10)$ group has 10 elements $((10(k-1)+1)_{th}, ..., 10k_{th})$ sampled from

Gaussian distribution $\mathcal{N}(\mathbf{1}, \delta\mathbf{I})$, and the rest being zeros. Then each column of X is perturbed by a random Gaussian noise ($\mathcal{N}(\mathbf{0}, \epsilon\mathbf{I})$). Our goal is to build a kernel matrix which is not affected by noises, i.e., a block diagonal symmetric matrix. Therefore, we compare the linear kernel and its tapered kernel. The constructed kernel matrices are plotted in Figure 6.1. It is clearly seen that the tapered kernel can significantly remove the noises from data, indicating good potentials for improved performance of kernel learning algorithms.

Here we evaluate the tapered kernels in the setting of two kernel based algorithms: kernel k-means and kernel NMF. Specifically, the kernel based algorithms equipped with tapered kernels are compared to those with corresponding traditional kernels.

6.3.1 Experimental settings

To evaluate our scheme, tapered kernel k-means and tapered kernel NMF are tested on the task of image clustering. The traditional kernel k-means and kernel NMF are treated as baselines, respectively. Note, k-means and NMF usually have different application and different merits, and here we are not interested in comparing k-means with NMF. Instead, we restrict our comparison to the kernel algorithms with traditional and tapered kernels.

For traditional kernels, three kernels are adopted: 1, linear kernel; 2, RBF kernel; 3, exponential kernel. As a result, the resulting tapered kernels are: 1, tapered linear kernel; 2, tapered RBF kernel; 3, tapered exponential kernel. So, both of the two algorithms (kernel k-means and kernel NMF) are tested with each of the six kernels. Spherical taper is used in our experiments for tapering.

The experiments are conducted on two data sets: Columbia Object Image Library (COIL-20) [75] and USPS data set. COIL-20 contains a set of gray-scale images of 72 different poses of 20 objects with black background. The USPS digit dataset contains gray scale images of hand written digits from 0 through 9, each image is of size 16×16, and there are 1100 images for each class.

When the clustering experiments are conducted, for COIL-20 database, all the 1440 samples are used for all experiments, and 20 class clustering is conducted, for USPS data set, each time 1000 samples are randomly selected, and then 10 class clustering is conducted. The parameter θ in taper is set to 1 for COIL-20, and 2 for USPS.

For evaluation, accuracy and Normalized Mutual Information (NMI) are adopted as the criteria to measure the clustering performance of the algorithms involved [53]. For both of criteria, a larger value means better performance of clustering. To reduce the effect of randomness induced by the non-convexity in optimization and the sampling of images in experiments of USPS data set, all the experiments are repeated 10 times, and the average results (accuracy/NMI) are reported.

Accuracy: Assume the clustering algorithm is tested on a set of N samples $\{x_i | i = 1, \ldots, N\}$. Let r_i denote the cluster label of sample x_i and t_i the ground truth label. The definition of accuracy is:

$$accuracy = \frac{\sum_{i=1}^{N} \delta(t_i, map(r_i))}{N}, \tag{6.11}$$

where $map(x)$ performs the best permutation mapping from clusters to predicted labels by Kuhn-Munkres algorithm [55], and $\delta(x, y)$ is 1 if x is equal to y, and 0 otherwise. Accuracy is higher when more labels are predicted correctly.

NMI: NMI compares the cluster centers of the ground truth, which are denoted by C, and the ones predicted by algorithm, which are denoted by C', and is defined as follows:

$$MI(C, C') = \sum_{c \in C; c' \in C'} p(c, c') log \frac{p(c, c')}{p(c)p(c')}, \tag{6.12}$$

where $p(c, c')$ denotes the joint probability that a sample belongs to cluster c and cluster c' while $p(c)$ and $p(c')$ are the probabilities that a sample belongs to cluster c and c' respectively. Let $H(C)$ and $H(C')$ be the entropies of C and C', then the normalized mutual information (NMI) is:

$$NMI(C, C') = \frac{MI(C, C')}{max((H(C)), (H(C')))}. \tag{6.13}$$

Basically, NMF measures the similarity of two distributions.

6.3.2 Experimental results

Tapered kernel algorithms are compared with corresponding traditional kernel algorithms. Clustering results are listed in tables, the winner is in bold.

Kernel k-means with three different kernels are compared with corresponding tapered kernel k-means on the data set COIL-20 in Table 6.1. We can easily see that tapered kernel k-means have better performance than traditional kernel ones, especially for linear kernel and RBF kernel. The performance of tapered kernel k-means on data set USPS is shown in Table 6.2. Tapered one again has either better or nearly equivalent (with difference ≤ 0.01) performance in the three cases.

Table 6.1: Results of (tapered) kernel k-means on COIL-20.

Methods	accuracy	NMI
linear kernel	0.546	0.689
tapered linear kernel	**0.601**	**0.736**
RBF kernel	0.578	0.713
tapered RBF kernel	**0.595**	**0.738**
exponential kernel	0.573	0.721
tapered exponential kernel	**0.603**	**0.732**

Table 6.2: Results of (tapered) kernel k-means on USPS.

Methods	accuracy	NMI
linear kernel	0.648	0.647
tapered linear kernel	**0.682**	**0.683**
RBF kernel	**0.707**	0.679
tapered RBF kernel	0.697	**0.692**
exponential kernel	**0.679**	0.677
tapered exponential kernel	0.672	**0.679**

The results of kernel NMF and tapered kernel NMF on COIL-20 and USPS are shown in Table 6.3 and Table 6.4, respectively. Again, we can easily see that the tapered kernel algorithms outperform the traditional kernel NMF in 11 out of 12 competitions.

Table 6.3: Results of (tapered) kernel NMF on COIL-20.

Methods	accuracy	NMI
linear kernel	0.355	0.464
tapered linear kernel	**0.591**	**0.714**
RBF kernel	0.453	0.575
tapered RBF kernel	**0.594**	**0.718**
exponential kernel	0.491	0.596
tapered exponential kernel	**0.552**	**0.697**

Table 6.4: Results of (tapered) kernel NMF on USPS.

Methods	accuracy	NMI
linear kernel	0.563	0.515
tapered linear kernel	**0.612**	**0.596**
RBF kernel	0.604	0.572
tapered RBF kernel	**0.632**	**0.620**
exponential kernel	**0.569**	0.535
tapered exponential kernel	0.568	**0.585**

From the results, we can safely conclude that the tapered kernels result in better performance in terms of both accuracy and NMI. We believe that this boosting in performance benefits from the tapering technique, which introduces sparseness to the kernel matrix while preserving most of the useful information.

Though the main reason to introduce taper into kernel is that the real relation among images should be sparse, there is a side effect that the tapered kernels make the related algorithms more effective in terms of both time and space. Specifically, it saves space due to the sparsity of the tapered kernels. Also, for the same reason, the computational cost will be lower than traditional dense kernel. For example, kernel k-means with tapered linear kernel on COIL-20 takes about 0.73 seconds, while the one with untapered linear kernel costs 1.07 seconds. Here about 30% of time is saved. The algorithm can be even more efficient if we carefully design it and specifically take advantage of the sparsity to speed up.

7 PIECEWISE LINEAR NONNEGATIVE MATRIX FACTORIZATION

The rapidly increasing large scale data pose challenges to the storage and computational resources, and make many computer vision and pattern recognition tasks prohibitively expensive. Dimension reduction techniques explore hidden structures of the original high dimensional data and learn new low dimensional representation to alleviate the challenges, and thus they are important for modelling relation among images. Popular dimension reduction techniques, such as PCA and NMF, do an efficient linear mapping to low dimensional space, while nonlinear techniques overcomes the limitation of linearity at the cost of expensive computational cost (e.g. computing the pairwise distance to find the geodesic distance). In this chapter, a piecewise linear dimension reduction technique with global consistency and smoothness constraint is proposed to overcome the restriction of linearity at relatively low cost. Extensive experimental results show that the proposed methods outperform the linear method in the scenario of clustering both consistently and significantly.

7.1 Introduction

Dimension reduction is playing an increasingly important role in many computer vision and pattern recognition tasks due to the rapidly growing large scale data with high dimensionality. It reduces the number of random variables under consideration, which not only saves computational and storage resources but also helps overcome the curse of dimensionality.

To explore the hidden structures of the original high dimensional data samples, $\{x^i = (x_1^i, \ldots, x_m^i)^\top | i = 1, \ldots, n\}$, dimension reduction techniques find low dimensional representation, $\{y^i = (y_1^i, \ldots, y_l^i)^\top | i = 1, \ldots, n\}$, where $l < m$, according to some certain criterion to capture the content in original data. The algorithms may be either linear or nonlinear. Typical linear methods, including principal component analysis (PCA), independent component analysis (ICA), nonnegative matrix factorization (NMF) and etc., are usually simple and efficient, while nonlinear dimensionality reduction techniques, such as locally linear embedding (LLE), Isomap, kernel tricks and Laplacian eigenmaps, allow nonlinearity during the dimension reduction to capture more property of data, for instance, underlying manifold structure, and thus to overcome the limitation of linear models. However, nonlinear ones are usually more computationally expensive, sometimes prohibitively.

NMF has been widely used in applications such as classification and clustering due to its easy theoretical interpretation and desired practical performance. It aims to approximate nonnegative high dimensional data by product of a low-rank basis matrix and another low-rank coefficient matrix, both of which are nonnegative. Variants of NMF algorithms are proposed to adapt to different situations. Sparse solutions of NMF are gained by adding extra sparseness regularization [27, 46]. Discriminative NMF algorithm maximizes the between-class distance and minimizes the within-class distance while learning the low dimensional representation [47, 48]. Research works in [17, 28] have proposed NMF variants on manifolds by adding constraints on local structures, since high dimensional data of many applications are on low dimensional manifold [24, 49].

However, all these methods are restricted to linear dimension reduction, and thus unable to capture complex nonlinear properties. Moreover, many real world data require nonlinearity in dimension reduction due to their distribution [16], for example, handwritten digits form own manifolds in the feature space. Unfortunately, typical nonlinear dimension reduction techniques are prohibitively expensive when facing large data. For example, Isomap has to compute the pairwise distances between sample to find geodesic distance, kernel trick has to apply the kernel function to all pairs of samples, etc.

To allow nonlinearity in the dimension reduction and to keep the computational efficiency, a piecewise linear dimension reduction technique is proposed in this chapter. Specifically, piecewise linear nonnegative matrix factorization (PLNMF) is proposed. The assumption is that though dimension reduction is nonlinear globally, it is linear locally, i.e. linear dimension reduction method is applicable when data are restricted to a region. As illustrated in Figure 7.1, the entire space is divided into 3 regions, within each of which a local model is learned. A global nonlinear dimension reduction is composed of a collection of linear local dimension reduction models. The proposed piecewise linear algorithm has been evaluated on eight real world datasets, and it has been shown to generate more accurate results than the linear model, NMF, in the scenario of clustering.

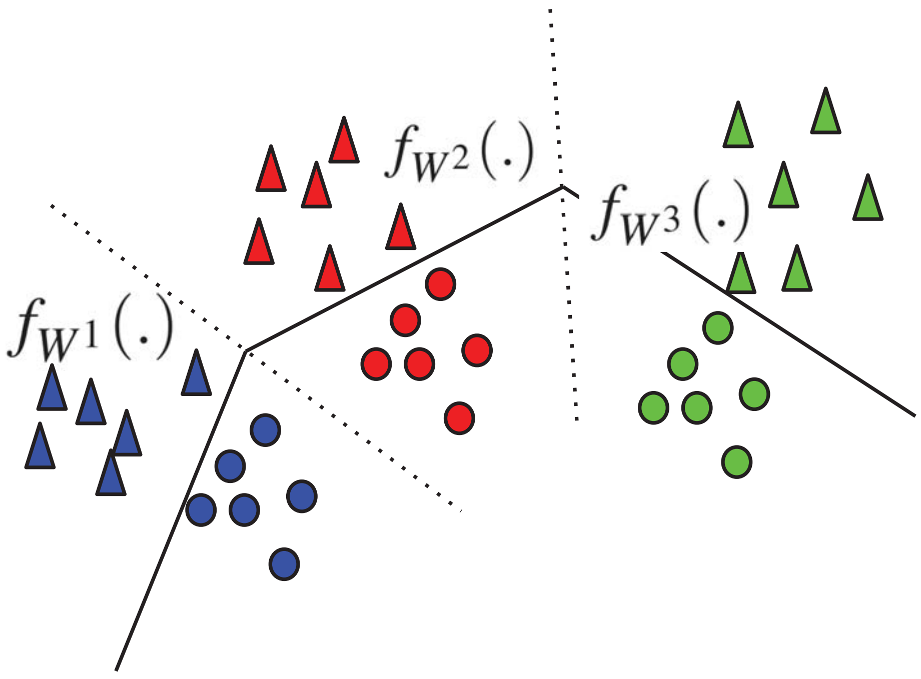


Figure 7.1.: Illustration of piecewise linearity: different colors correspond to different regions according to the data partition. Different shapes are for data samples in different classes. A local model $f_{W^k}(.)$ is learned from samples within each region.

7.2 Piecewise Linear Nonnegative Matrix Factorization

Given a nonnegative data matrix $X \in \mathbb{R}_+^{m\times n}$, each column of which is a data sample of dimension m, NMF seeks two nonnegative matrices $W \in \mathbb{R}_+^{m\times p}$ and $H \in \mathbb{R}_+^{p\times n}$, where p is usually much smaller than both m and n and controls the ranks of matrices W and H, by minimizing a loss function $\mathcal{L}(X, WH)$. A typical choice is the following

$$
\begin{aligned}
\mathcal{L}(X, WH) =& \sum_{i=1}^{n} \ell(x^i, Wh^i) \\
=& ||X - WH||_F^2,
\end{aligned}
\tag{7.1}
$$

where $\ell(x, y) = \|x - y\|_2^2$ and $||.||_F$ denotes the Frobenius norm. Here W is treated as basis, and each column of H is new low dimensional representation of each column of X in new space.

Although the above objective function is nonconvex with respect to W and H jointly, it is convex with respect to either W or H. An alternative minimization algorithm for optimizing this objective function is to repeat the following two update rules [8]:

$$
W_{ij} = W_{ij} \frac{(XH^\top)_{ij}}{(WHH^\top)_{ij}}, \tag{7.2}
$$

$$
H_{ij} = H_{ij} \frac{(W^\top X)_{ij}}{(W^\top WH)_{ij}}. \tag{7.3}
$$

This section describes the proposed piecewise linear nonnegative matrix factorization (PLNMF), which provides locality sensitive dimension reduction. Figure 7.1 illustrates the idea of piecewise linearity, which is composed of two steps: 1, the data are partitioned into groups in a locality sensitive way; 2, local model is trained for samples within each of the groups with necessary regularization, which will be detailed later.

7.2.1 Locality sensitive data partition

The key idea is to partition samples into different groups such that samples in the same group are much closer to each other than samples in different groups. There are various approaches to achieve this, for example, 1, locality-sensitive hashing, which assigns locality sensitive numbers to samples by a hashing function such that samples with the same number are put in the same group; 2, k-means clustering, which groups samples into k clusters with each sample belonging to the cluster with the closest mean. By either of these two methods, a group ID, $g(x^i)$, is gained for each sample x^i. Say samples are partitioned into K groups, then $g(x^i) \in \{1, 2, \ldots, K\}, \forall i$. Then the matrix X formed by samples $\{x^i \in \mathbb{R}^m_+ | i = 1, \ldots, n\}$ can be represented as a concatenation of blocks, i.e. $X = [X^1, X^2, \ldots, X^K]$, with $X^k \in \mathbb{R}^{m \times n_k}_+$ corresponds the samples with ID equalling to k, where n_k is the number of columns in block X^k.

7.2.2 Local linear dimension reduction

Let f be a linear dimension reduction function, i.e.

$$f : \mathbb{R}^m \mapsto \mathbb{R}^p, \tag{7.4}$$

where $p < m$. When $p = 1$, it can be viewed as a typical linear regressor or classifier depends on the type of output value.

The function f is defined globally in the feature space, and it linearly projects a sample from m dimensional feature space to a low p dimensional space. The linearity of the projection usually ensures the simplicity and efficiency at the cost of flexibility, while a nonlinear model is usually capable of depicting more complex data distribution at the cost of high computational resources. For example, locality preserving projection (LPP) [76], is able to reduce dimension of samples drawn from a manifold, however, it requires to compute the pairwise distances of all samples, which may be prohibitively expensive especially when facing large data. Considering all the facts above, a model that allows flexibility while maintaining the efficiency is desired. Thus, we consider a piecewise linear model.

Assume the entire feature space is divided into regions $\{R_k | k = 1, \ldots, K\}$. For each region, there is a linear model trained locally and specifically. Denote the model for region R_k by f_k, then for an arbitrary sample, the model is

$$f(x) = \sum_{k=1}^{K} f_k(x) \mathcal{I}(x \in R_k), \tag{7.5}$$

where f_k is a linear dimension reduction model, i.e., $f_k : \mathbb{R}^m \mapsto \mathbb{R}^p$, and is trained within each region R_k, and $\mathcal{I}$ is indicator function.

7.2.3 Assembling local linear models

Unlike the the conventional NMF, which considers a global loss function, PLNMF aims to minimize the loss function defined on local regions $\{R_k | k = 1, \ldots, K\}$ that are determined by either LSH or k-means clustering. In PLNMF, sample x belongs to region R_k only if $g(x) = k$, i.e. $\mathcal{I}(x^i \in R_k) = \mathcal{I}(g(x^i) = k)$. Also, local model, which is determined by W^k, is

$$f_{W^k}(s) = \arg\min_h \|s - W^k h\|_2^2 \tag{7.6}$$

Accordingly, the piecewise linear model is gained by gluing all the local models together:

$$\begin{aligned} f(x) &= \sum_{k=1}^{K} \mathcal{I}(g(x) = k) \arg\min_h \ell(x^i, W^k h^i) \\ &= \sum_{k=1}^{K} \mathcal{I}(g(x) = k) \arg\min_h \|s - W^k h\|_2^2, \end{aligned} \tag{7.7}$$

To learn all the local models $\{W_k | k = 1, \ldots, K\}$, the loss function is defined as

$$\begin{aligned} \mathcal{L} &= \sum_{k=1}^{K} \sum_{i=1}^{n} \mathcal{I}(g(x^i) = k) \ell(x^i, W^k h^i) \\ &= \sum_{k=1}^{K} \|X^k - W^k H^k\|, \end{aligned} \tag{7.8}$$

where $W^k \in \mathbb{R}_+^{m\times p}, H^k \in \mathbb{R}_+^{p\times n_k}$. According to the formulation above, the partition of the data divided the feature space $\mathcal{S}$ into a collection of disjoint K regions, and the samples within the same region are handled by the same model W^k.

7.2.4 Local models with global regularizer

In NMF, model W determines the dimension reduction. Let f_W denote the model, i.e. $f_W : R_+^m \mapsto R_+^p$. For an arbitrary sample x in region k, i.e. $g(x) = k$, its p dimensional representation h is determined by f_{W^k}.

Unlike a classifier or regressor, which outputs a single value, we are usually interested in predicting more than one values. For instance, we want to predict the $temperature$ and $humidity$ next day, thus $p = 2$. Let $h = [h_1, h2]^T$ be the output. It is undesirable that h_1 output by model f_{W^i} means $temperature$ and the h_2 out by a different model f_{W^j} represents $temperature$. Unfortunately, this inconsistency happens easily if the local models are trained on individual regions independently. Note that W^k itself will be be learned from the samples in region k, X^k, by minimizing the objective function $\|X^k - W^k H^k\|$ with respect to both W^k and H^k. It is not difficult to verify that a simple column-wise permutation of W^k and a corresponding row-wise permutation of H^k will introduce a trivial different solution, and this permutation easily makes the interpretation of the same dimension of h inconsistent.

Thus it is necessary to ensure the consistency from different local models before collecting the set of models $\{f_{W^k} | k = 1, \dots, K\}$ to form the model for the global feature space.

To achieve this goal, a global reference model W, which is defined in the entire space, is introduced to regularize all local models. The regularizer is defined as

$$\mathcal{R}_1(W, \{W^k\}) = \sum_{k=1}^{K} ||W^k - W||_F^2 \tag{7.9}$$

To ensure the validity of the global reference model, we also introduce a regularizer so that the model W is able to capture property of all original samples in different regions. This regularizer is defined as

$$\mathcal{R}_2(W) = ||X - WH||_F^2 \tag{7.10}$$

Combining the regularizers with the loss function, the overall objective function can be written as:

$$\begin{aligned}\mathcal{O} =& \mathcal{L} + \alpha\mathcal{R}_1(W, \{W^k\}) + \beta\mathcal{R}_2(W)\\ =& \sum_{i=1}^{K} ||X^i - W^i H^i||_F^2 + \alpha \sum_{i=1}^{K} ||W^i - W||_F^2\\ &+ \beta||X - WH||_F^2\end{aligned} \tag{7.11}$$

α and β are parameters that control the strength of regularization. Specifically, α control the consistency and smoothness as will be discussed below. When α is 0, the formulation is factorized into a set of independent NMF problems on different regions according to the partition. When α is infinitely large, it reduces to a global NMF, since all the local models are forced to be exactly the same as the global one. By doing this, the proposed model is equipped with both locality characteristic and generalization ability on global feature space.

7.2.5 Discussion

Though the global reference model W is introduced mainly to disambiguate the inconsistency of different local models, it also helps smooth the final piecewise linear model and avoid potential overfitting.

The regularity and smoothness is not guaranteed while gluing a collection of independently trained local models $\{W^k | k = 1, \ldots, K\}$ into a global one, since individual model W^k is limited on a local region. Specifically, for example, two neighboring samples x^i and x^j from two different regions have low dimensional representation $h_i = \arg\min_h \|s - W^g(x^i)h\|_2^2$ and $h_j = \arg\min_h \|s - W^g(x^j)h\|_2^2$, the difference between them may be very large. With global regularizer the difference will be reduced.

Without a global regularizer, the piecewise linear model factorizes into independent models defined on different local regions. The learned local models have overfitting risk due to relatively small amount of samples in each individual region. By introducing the global reference model, the information of samples in one local region will propagate to other regions to help relieve the overfitting effect and improve the robustness of the piecewise linear model.

7.3 Optimization

In this section, we discuss the optimization of the objective function in Equation 7.11. Obviously, it is not convex with respect to $\{W^k | k = 1, \ldots, K\}$, $\{H^k | k = 1, \ldots, K\}$, W and H, jointly. Thus, an alternative optimization technique is adopted. A subset of variables, say W, is being optimized while the others are fixed. Different variables are optimized in turn until convergence. Specifically, we need to optimize the objective function with respect to $W, H, \{W^k | k = 1, \ldots, K\}$ and $\{H^k | k = 1, \ldots, K\}$ alternatively. The procedures are detailed in the following subsections.

7.3.1 Optimize local models and their coefficients

First, we describe how to optimize the local models and corresponding coefficients while keeping global model W and new low dimensional representation H fixed. Fortunately, when W and H are fixed, the optimization problem factorizes into a set of independent subproblems, each of which involves a single local model W^k and its corresponding coefficient matrix H^k. Thus, it is sufficient to solve these subproblems independently. Specifically, for an arbitrary W^k, the subproblem is

$$\begin{aligned}\arg\min_{W^k} \mathcal{O} &= \arg\min_{W^k} \sum_{i=1}^{K} ||X^k - W^k H^k||_F^2 \\ &\quad + \alpha \sum_{k=1}^{K} ||W^k - W||_F^2 + \beta ||X - WH||_F^2 \\ &= \arg\min_{W^k} ||X^k - W^k H^k||_F^2 + \alpha ||W^k - W||_F^2 \\ &= \arg\min_{W^k} ||[X^k, \sqrt{\alpha} W] - W^k [H^k, \sqrt{\alpha} I]||_F^2\end{aligned} \tag{7.12}$$

Let $\tilde{X} \doteq [X^k, \sqrt{\alpha}W]$ and $\tilde{H} \doteq [H^k, \sqrt{\alpha}I]$. The problem reduces to the conventional NMF problem. Following the conventional NMF, and the objective function will decrease when W^k is updated as

$$W_{ij}^k = W_{ij}^k \frac{(\tilde{X}\tilde{H}^\top)_{ij}}{(W^k \tilde{H}\tilde{H}^\top)_{ij}}. \tag{7.13}$$

Similarly, the optimization with respect to H^k can be achieved as

$$\begin{aligned}\arg\min_{H^k} \mathcal{O} &= \arg\min_{H^k} \sum_{i=1}^{K} ||X^k - W^k H^k||_F^2 \\ &\quad + \alpha \sum_{k=1}^{K} ||W^k - W||_F^2 + \beta ||X - WH||_F^2 \\ &= \arg\min_{H^k} ||X^k - W^k H^k||_F^2\end{aligned} \tag{7.14}$$

From the objective function, it is not difficult to see that H^k only depends on W^k when W^k is given. Updating of H^k is the same as standard NMF:

$$H_{ij}^k = H_{ij}^k \frac{({W^k}^\top X^k)_{ij}}{({W^k}^\top W^k H^k)_{ij}}. \tag{7.15}$$

7.3.2 Optimize global model and its coefficients

The global reference model W serves as regularizer to ensure the consistency and s-moothness of local models, and it is also learned from data. The optimization of W involves the two regularizers in the objective function.

$$\begin{aligned}\arg\min_W \mathcal{O} &= \arg\min_W \sum_{i=1}^{K} ||X^k - W^k H^k||_F^2 \\ &\quad + \alpha \sum_{k=1}^{K} ||W^k - W||_F^2 + \beta ||X - WH||_F^2 \\ &= \arg\min_W \alpha \sum_{k=1}^{K} ||W^k - W||_F^2 + \beta ||X - WH||_F^2 \\ &= \arg\min_W ||[\sqrt{\alpha}W^1, \ldots, \sqrt{\alpha}W^K, \sqrt{\beta}X] \\ &\quad - W[\sqrt{\alpha}I, \ldots, \sqrt{\alpha}I, \sqrt{\beta}H]||_F^2\end{aligned} \tag{7.16}$$

Let $\tilde{X} \doteq [\sqrt{\alpha}W^1, \ldots, \sqrt{\alpha}W^K, \sqrt{\beta}X]$ and $\tilde{H} \doteq [\sqrt{\alpha}I, \ldots, \sqrt{\alpha}I, \sqrt{\beta}H]||_F^2$. To decrease the objective function, we update W by

$$W_{ij} = W_{ij} \frac{(\tilde{X}\tilde{H}^\top)_{ij}}{(W\tilde{H}\tilde{H}^\top)_{ij}}. \tag{7.17}$$

The optimization of H only involves the second regularizer. Thus, other terms can be discarded during the optimization, and the problem reduces to $\arg\min_H \mathcal{O} = \arg\min_H ||X - WH||_F^2$. The updating rule of H is the same as rule in Equation 7.3.

7.3.3 Overall algorithm

The overall algorithm for piecewise linear nonnegative matrix factorization, which includes initialization and iterative optimization, is summarized in Algorithm 5. The stopping criteria is determined by setting both a maximum number of iterations and a tolerance threshold ϵ. The convergence of the iterative optimization is guaranteed since all of the

Table 7.1: Dataset statistics.

Datasets	$\sharp Classes$	$\sharp Samples$	$\sharp Attributes$	$Other\ description$
ORL	40	400	1024	Each class has 10 images.
USPS	10	11000	256	Each class has 1100 images.
COIL20	20	1440	1024	Each class has 72 samples.
Yale	15	165	1024	Each subject has 11 images.
Yale B	38	2414	1024	Each subject has around 64 images.
Wine	3	178	13	
Leaf	40	340	14	
Glass	7	214	9	

Algorithm 5 Piecewise linear nonnegative matrix factorization

```
Require: Samples $\{x^i|i = 1, \dots, n\}$, $K$, $p$, $\alpha$ and $\beta$
 1: Partition the data samples into $K$ groups and form matrix $X = [X^1, \dots, X^K] \in \mathbb{R}_+^{m\times n}$.
 2: Randomly initialize $W \in \mathbb{R}_+^{m\times p}$, $H \in \mathbb{R}_+^{p\times n}$, $W^k \in \mathbb{R}_+^{m\times p}$ and $H^k \in \mathbb{R}_+^{p\times n_k}$, $\forall k$.
 3: while not converge do
 4:     for $k = 1 : 1 : K$ do
 5:         Update $W^k$ according to Equation 7.13
 6:     end for
 7:     for $k = 1 : 1 : K$ do
 8:         Update $H^k$ according to Equation 7.15
 9:     end for
10:     Update $W$ according to Equation 7.17
11:     Update $H$ according to Equation 7.3
12: end while
13: return $W$, $H$, $\{W^k|k = 1, \dots, K\}$ and $\{H^k|k = 1, \dots, K\}$
```

Table 7.2: Performance comparison on various datasets.

Datasets	NMF	$PLNMF$
ORL	40.18 ± 2.37	$\mathbf{41.83 \pm 2.05}$
USPS	51.46 ± 5.12	$\mathbf{65.28 \pm 4.65}$
COIL20	46.14 ± 2.78	$\mathbf{53.06 \pm 2.89}$
Yale	34.55 ± 2.23	$\mathbf{38.24 \pm 2.17}$
Yale B	19.46 ± 0.70	$\mathbf{20.53 \pm 1.58}$
Wine	47.02 ± 7.16	$\mathbf{55.96 \pm 6.13}$
Leaf	26.18 ± 2.03	$\mathbf{31.06 \pm 3.00}$
Glass	27.20 ± 2.72	$\mathbf{41.17 \pm 3.14}$

updating rules involved decrease the objective function [8] and the objective function is lower bounded obviously, such as 0.

7.4 Experiments

To evaluate the performance of the proposed piecewise linear nonnegative matrix factorization, we conduct experiments on benchmark datasets. By designing the experiments, we aim to answer the question whether or not the flexility induced by using a set of local linear models instead of one will help in real tasks, such as clustering. The detailed experimental settings and results are shown in the remaining of this section.

7.4.1 Experimental settings

In our experiments, the locality sensitive data partition is conducted by k-means clustering. The number of partitions, K, is set to 2 for all datasets, and we do not tune K across different datasets. Parameter α and β are searched on grid $\{10^{-4}, 10^{-3}, \dots, 10^{4}\}$. The performance with different values of α and β are also reported.

To quantitatively compare the performance of dimension reduction methods, we conduct clustering experiments on a collection of benchmark datasets. Each dataset is composed of samples from two or more classes. For a dataset having c classes of samples, the parameter p is set the same as c, and the clustering is conducted by assigning each sample to the cluster corresponding to the maximum component of its p dimensional representation.

Accuracy metric [53, 54] is used for evaluation of the clustering. For a sample $x^i,\ i \in \{1, \dots, n\}$, let r_i denote the cluster label and t_i denote the ground true label. The accuracy is defined as follows:

$$accuracy = \frac{\sum_{i=1}^{n} \delta(t_i, map(r_i))}{n}, \tag{7.18}$$

where $\delta(x, y)$ is 1 if x is equal to y, and 0 otherwise. Function $map(.)$ denotes the best permutation mapping function by Kuhn-Munkres algorithm [55], which maps each cluster to the corresponding predicted label. Thus, the more labels of samples are predicted correctly, the higher the accuracy is.

Considering that the overall optimization problem is nonconvex, it is difficult to achieve global optimum and different initializations may result in different solutions. To make the experiments statistical meaningful, all the experiments are repeated by 10 times, and the mean values and standard deviations are reported.

Experiments are conducted on various datasets, including ORL, USPS, COIL20, Yale, Yale B, Wine, Leaf, and Glass. ORL, Yale and Yale B are face datasets, where the image are cropped and resized to 32×32. USPS is handwritten digit database. COIL20 is composed of multiple view images of 20 objects. The last three datasets are from UCI machine learning data repository.

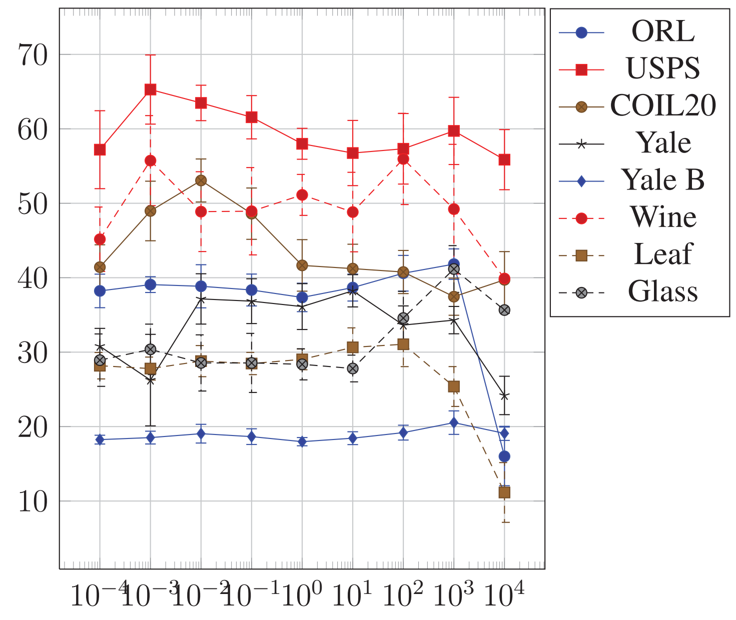
Each dataset has samples from different categories. Detailed data statistics are shown in Table 7.1. When we use dataset USPS, 100 samples per class are randomly selected due to the relatively large size. For other datasets, all samples are used for clustering.

7.4.2 Experimental results

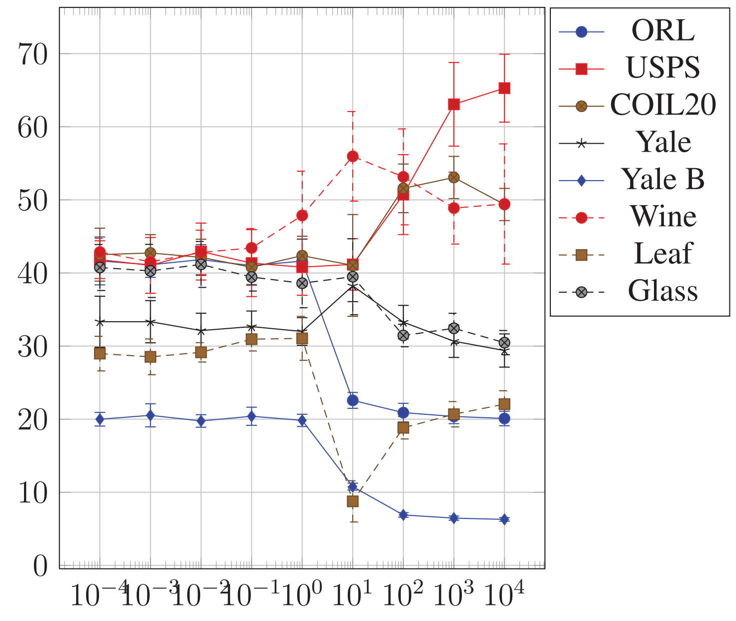
The accuracy of clustering results on benchmark datasets are reported in Table 7.2. For all the eight datasets, the proposed PLNMF achieves significantly higher clustering accuracy than conventional NMF. For USPS and glass datasets, PLNMF is about 14% better than NMF. For datasets COIL20, Wine and Leaf, the advantage is about 5% or higher. For the other three face datasets, the proposed PLNMF perform also slightly better. We believe the superiority comes from the adaptiveness to local regions of PLNMF.

Interestingly, the advantage of proposed PLNMF on datasets USPS, COIL20, Wine, Leaf, and Glass is more significant than on the three face datasets: ORL Yale and Yale B. Checking the detailed distribution of those datasets, it is not difficult to find that the first five datasets are more diverse than the face datasets. Intuitively, the more diverse the dataset is, the more complex model it requires to fit. Thus, PLNMF has more space to improve the clustering performance for these five datasets.

We also study the sensistivity of the performance on the parameters α and β. The clustering accuracy with varying values of α and β is shown in Figure 7.2. Note that the x-axis in Figure 7.2 is logarithmic. When α lies in the range $[10^{-4}, 10^{3}]$, the clustering accuracy of PLNMF is relatively stable for all datasets. For β, the accuracy for most datasets are stable and good for $\beta \leq 1$. Three datasets (USPS, Wine and COIL20) achieve optimal performance when $\beta \geq 10$. However, for each individual pdataset, the sensitivity to β is



(a) Clustering accuracy(%) with parameter α



(b) Clustering accuracy(%) with parameter β

Figure 7.2.: Parameter sensitivity of LPNMF.

still low. Thus, we can safely reach the conclusion that the proposed PLNMF outperforms conventional NMF, and the performance of PLNMF is reasonably insensitive to parameters.

8 CONCLUSIONS AND FUTURE DIRECTIONS

8.1 Conclusions

In summary, this dissertation conducts preliminary research on the following important issues in image relation modelling as well as its applications and related optimization.

Misalignment: Images are often loosely or not controlled in real-world applications, and as a result misalignment is pervasive and impairs relation modelling, including factorization and classification. This dissertation extends the existing NMF and SVM algorithms by integrating aligning process seamlessly to achieve insensitivity to misalignment.

Multi-manifold structure: In many real situations especially when data are in high dimensional space, samples typically lie on low dimensional manifolds of the feature space. The structure of underlying manifolds has been proven to be critical for the performance of many image processing tasks. This dissertation develops factorization and classification algorithms which explore the sparse structures on multiple manifolds.

Challenges in nonlinearity: Modelling relation among large-scale complex data requires the flexibility of nonlinearity, which may be introduced by kernel trick or piecewise linearity. This dissertation introduces an effective simple approach to sparse kernels for image processing. Also, a piecewise linear factorization algorithm with global consistency and smoothness constraint is proposed to enjoy both the efficiency of the linear model and the power of nonlinearity in dimension reduction.

The aforementioned issues are critical to many applications, including, but not limited to, image clustering and classification. This dissertation makes attempt to address these issues in real applications and opens up opportunities for future work.

8.2 Future Directions

Most of existing modelling techniques rely on batch rather than online processing. However, online algorithms will not only satisfy the requirements of real online input such as video stream, but also bring potential efficiency and scalability to applications of image relation for large-scale data.

Nowadays, combining with the popularity with visual data, there come other information resources, which have the potential to help the modelling of image relation. For example, the context of a social image will provide side information such as tags, ratings and location information. Prior knowledge on side information could be easily adopted, such as images with the same tags tend to have similar contents. Hence modelling images relation should go beyond content and incorporate more information in real applications.

LIST OF REFERENCES

LIST OF REFERENCES

[1] Grigorios Tzortzis and Aristidis Likas. The global kernel k-means clustering algorithm. In *IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 2008.

[2] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[3] Bernhard Schölkopf and Alexander J Smola. *Learning with Kernels*. The MIT Press, 2002.

[4] Daoqiang Zhang, Zhi-Hua Zhou, and Songcan Chen. Non-negative matrix factorization on kernels. *PRICAI 2006: Trends in Artificial Intelligence*, 4099:404–412, 2006.

[5] Jian Yang, Alejandro F Frangi, Jing-yu Yang, David Zhang, and Zhong Jin. KPCA plus LDA: A complete kernel fisher discriminant framework for feature extraction and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(2):230–244, 2005.

[6] Bernhard Scholkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *Artificial Neural Networks–ICANN'97*, pages 583–588. Springer, 1997.

[7] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means: spectral clustering and normalized cuts. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.

[8] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems*, 2001.

[9] Junzhou Huang, Xiaolei Huang, and D. Metaxas. Simultaneous image transformation and sparse representation recovery. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[10] Erik G Learned-Miller. Data driven image models through continuous joint alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(2):236–250, 2006.

[11] Mark Cox, Sridha Sridharan, Simon Lucey, and Jeffrey Cohn. Least squares congealing for unsupervised alignment of images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[12] Yigang Peng, Arvind Ganesh, John Wright, Wenli Xu, and Yi Ma. RASL: Robust alignment by sparse and low-rank decomposition for linearly correlated images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[13] Julian Eggert, Heiko Wersing, and E Korner. Transformation-invariant representation and NMF. In *IEEE International Joint Conference on Neural Networks*, 2004.

[14] Leah Bar and Guillermo Sapiro. Hierarchical dictionary learning for invariant classification. In *IEEE International Conference on Acoustics Speech and Signal Processing*, 2010.

[15] Jun He, Dejiao Zhang, L. Balzano, and Tao Tao. Iterative online subspace learning for robust image alignment. In *IEEE International Conference and Workshops on Automatic Face and Gesture Recognition*, 2013.

[16] Andrew Goldberg, Xiaojin Zhu, Aarti Singh, Zhiting Xu, and Robert Nowak. Multi-manifold semi-supervised learning. In *International Conference on Artificial Intelligence and Statistics*, 2009.

[17] Deng Cai, Xiaofei He, Xuanhui Wang, Hujun Bao, and Jiawei Han. Locality preserving nonnegative matrix factorization. In *International Joint Conference on Artifical Intelligence*, 2009.

[18] Quanquan Gu and Jie Zhou. Neighborhood preserving nonnegative matrix factorization. In *British Machine Vision Conference*, 2009.

[19] Vladimir Vapnik. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24:774–780, 1963.

[20] Vladimir Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

[21] Guodong Guo, Stan Z Li, and Kap Luk Chan. Face recognition by support vector machines. In *IEEE International Conference on Automatic Face and Gesture Recognition*, 2000.

[22] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627 –1645, 2010.

[23] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Machine Learning: ECML-98*, volume 1398 of *Lecture Notes in Computer Science*, pages 137–142. 1998.

[24] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[25] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[26] Patrik O Hoyer. Non-negative sparse coding. In *IEEE Workshop on Neural Networks for Signal Processing*, 2002.

[27] Jingu Kim and Haesun Park. Sparse nonnegative matrix factorization for clustering. In *CSE Technical Reports*. Georgia Institute of Technology, 2008.

[28] Bin Shen and Luo Si. Nonnegative matrix factorization clustering on multiple manifolds. In *AAAI Conference on Artificial Intelligence*, 2010.

[29] Chris Ding, Tao Li, and Michael I Jordan. Convex and semi-nonnegative matrix factorizations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):45–55, 2010.

[30] David Guillamet and Jordi Vitrià. Non-negative matrix factorization for face recognition. In *Catalonian Conference on AI*, 2002.

[31] Michael W Berry, Murray Browne, Amy N Langville, V Paul Pauca, and Robert J Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics & Data Analysis*, 52(1):155–173, 2007.

[32] Chih-Jen Lin. Projected gradient methods for nonnegative matrix factorization. *Neural Computation*, 19(10):2756–2779, 2007.

[33] Hyunsoo Kim and Haesun Park. Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. *SIAM Journal on Matrix Analysis and Applications*, 30(2):713–730, 2008.

[34] Cho-Jui Hsieh and Inderjit S Dhillon. Fast coordinate descent methods with variable selection for non-negative matrix factorization. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2011.

[35] Jae Sung Lee, Daniel D Lee, Seungjin Choi, Kwang Suk Park, and Dong Soo Lee. Non-negative matrix factorization of dynamic images in nuclear medicine. In *IEEE Nuclear Science Symposium Conference Record*, volume 4, pages 2027–2030, 2001.

[36] Gary B Huang, Marwan Mattar, Tamara Berg, Eric Learned-Miller, et al. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. In *Workshop on Faces in 'Real-Life' Images: Detection, Alignment, and Recognition*, 2008.

[37] Ralph Gross, Iain Matthews, Jeffrey Cohn, Takeo Kanade, and Simon Baker. Multi-PIE. *Image and Vision Computing*, 28(5):807–813, 2010.

[38] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[39] Jonathan J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, 1994.

[40] Olga Russakovsky, Yuanqing Lin, Kai Yu, and Li Fei-Fei. Object-centric spatial pooling for image classification. In *European Conference on Computer Vision*, 2012.

[41] Bao-Di Liu, Yu-Xiong Wang, Yu-Jin Zhang, and Bin Shen. Learning dictionary on manifolds for image classification. *Pattern Recognition*, 46(7):1879–1890, July 2013.

[42] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *The Journal of Machine Learning Research*, 7:2399–2434, 2006.

[43] Linli Xu, James Neufeld, Bryce Larson, and Dale Schuurmans. Maximum margin clustering. In *Advances in Neural Information Processing Systems*, 2004.

[44] John Wright, Allen Y Yang, Arvind Ganesh, Shankar S Sastry, and Yi Ma. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):210–227, 2009.

[45] Yi Wu, Bin Shen, and Haibin Ling. Visual tracking via online nonnegative matrix factorization. *IEEE Transactions on Circuits and Systems for Video Technology*, 24(3):374–383, March 2014.

[46] Patrik O. Hoyer. Non-negative matrix factorization with sparseness constraints. *The Journal of Machine Learning Research*, 5:1457–1469, 2004.

[47] Stefanos Zafeiriou, Anastasios Tefas, Ioan Buciu, and Ioannis Pitas. Exploiting discriminant information in nonnegative matrix factorization with application to frontal face verification. *IEEE Transactions on Neural Networks*, 17(3):683–695, 2006.

[48] Yuan Wang, Yunde Jia, Changbo Hu, and Matthew Turk. Fisher non-negative matrix factorization for learning local features. In *Asian Conference on Computer Vision*, 2004.

[49] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[50] David L. Donoho. For most large underdetermined systems of equations, the minimal l1-norm near-solution approximates the sparsest near-solution. *Communications on Pure and Applied Mathematics*, 59(7):907–934, 2006.

[51] Berwin A Turlach, William N Venables, and Stephen J Wright. Simultaneous variable selection. *Technometrics*, 47(3):349–363, 2005.

[52] Ehsan Elhamifar and René Vidal. Sparse subspace clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[53] Wei Xu, Xin Liu, and Yihong Gong. Document clustering based on non-negative matrix factorization. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, 2003.

[54] Deng Cai, Xiaofei He, and Jiawei Han. Graph regularized non-negative matrix factorization for data representation. In *UIUC Computer Science Research and Tech Reports*, 2008.

[55] László Lovász and Michael D Plummer. *Matching Theory*. Akademiai Kiado, 1986.

[56] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: primal estimated sub-gradient solver for svm. *Mathematical Programming*, 127(1):3–30, 2011.

[57] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

[58] Shin Matsushima, SVN Vishwanathan, and Alexander J Smola. Linear support vector machines via dual cached loops. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 177–185. ACM, 2012.

[59] David L Donoho. For most large underdetermined systems of linear equations the minimal l1-norm solution is also the sparsest solution. *Communications on Pure and Applied Mathematics*, 59(6):797–829, 2006.

[60] Emmanuel J Candes, Justin K Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59(8):1207–1223, 2006.

[61] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[62] John Platt et al. Sequential minimal optimization: A fast algorithm for training support vector machines. In *Technical Report MSR-TR-98-14*. 1998.

[63] Athinodoros S. Georghiades, Peter N. Belhumeur, and David Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):643–660, 2001.

[64] Aleix M Martinez. The AR face database. *CVC Technical Report*, 24, 1998.

[65] Terence Sim, Simon Baker, and Maan Bsat. The CMU pose, illumination, and expression (PIE) database. In *IEEE International Conference on Automatic Face and Gesture Recognition*, 2002.

[66] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *European Conference on Computer Vision*, 2010.

[67] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y Ng. Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems*, 2007.

[68] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27, 2011.

[69] Bin Shen, Wei Hu, Yimin Zhang, and Yu-Jin Zhang. Image inpainting via sparse representation. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2009.

[70] Cari G. Kaufman, Mark J. Schervish, and Douglas W. Nychka. Covariance tapering for likelihood-based estimation in large spatial data sets. *Journal of the American Statistical Association*, 103(484):1545–1555, 2008.

[71] Reinhard Furrer, Marc G Genton, and Douglas Nychka. Covariance tapering for interpolation of large spatial datasets. *Journal of Computational and Graphical Statistics*, 15(3), 2006.

[72] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012.

[73] Anil K Jain and Richard C Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., 1988.

[74] Radha Chitta, Rong Jin, Timothy C Havens, and Anil K Jain. Approximate kernel k-means: solution to large scale kernel clustering. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2011.

[75] Sammeer A Nene, Shree K Nayar, and Hiroshi Murase. Columbia object image library (COIL-20). *Dept. Comput. Sci., Columbia Univ., New York.[Online] http://www. cs. columbia. edu/CAVE/coil-20. html*, 62, 1996.

[76] Xiaofei He and Partha Niyogi. Locality preserving projections. In *Advances in Neural Information Processing Systems*, 2003.

VITA

VITA

Bin Shen was born in Jiaxing, Zhejiang Province, China. He earned the B.S. and M.S. degrees in 2007 and 2009, respectively, both from the Department of Electronic Engineering, Tsinghua University, Beijing, China. In 2009, he joined the Department of Computer Science at Purdue University, West Lafayette, Indiana, USA as a Ross fellow, where he earned another M.S. degree in 2011 and the Ph.D. degree in 2014.