Discriminative Linear and Multilinear Subspace Methods

Dacheng TAO

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of Doctor of Philosophy

Birkbeck
UNIVERSITY OF LONDON

October 2006

School of Computer Science and Information Systems

This thesis is submitted to the School of Computer Science and Information Systems, Birkbeck College, University of London in partial fulfilment of the requirements for the degree of Doctor of Philosophy. I hereby declare that this thesis is my own work and except where otherwise stated. This submission has not been submitted for a degree to any other university or institution.

Dacheng TAO
School of Computer Science and Information Systems
Birkbeck College
University of London

To my family.

Table of Contents

List of Figures

List of Tables

Abstract

Linear discriminant analysis (LDA) sheds light on classification tasks in computer vision. However, classification based on LDA can perform poorly in applications because LDA has: 1) the heteroscedastic problem, 2) the multimodal problem, 3) the class separation problem, and 4) the small sample size (SSS) problem. In this thesis, the first three problems are called the model based problems because they arise from the definition of LDA. The fourth problem arises when there are too few training samples. The SSS problem is also known as the overfitting problem.

To address the model based problems, a new criterion is proposed: maximization of the geometric mean of the Kullback–Leibler (KL) divergences and the normalized KL divergences for subspace selection when samples are sampled from Gaussian mixture models. The new criterion reduces all model based problems significantly, as shown by a large number of empirical studies.

To address the SSS problem in LDA, a general tensor discriminant analysis (GTDA) is developed. GTDA makes better use of the structure information of the objects in vision research. GTDA is a multilinear extension of a modified LDA. It involves the estimation of a series of projection matrices in projecting an object in the form of a tensor from a high dimensional feature space to a low dimensional feature space. Experiments on human gait recognition demonstrate that GTDA combined with LDA and nearest neighbor rule outperforms competing methods.

Based on the work above, the standard convex optimization based approach to machine learning is generalized to the supervised tensor learning (STL) framework, in which tensors are accepted as input. The solution to STL is obtained in practice using an alternating projection algorithm. This generalization reduces the overfitting problem when there are only a few training samples. An empirical study confirms that the overfitting is reduced.

Acknowledgement

Professor Stephen J. Maybank is an excellent advisor and friend! During the two years for my PhD study, I owed him a lot because I have obtained everlasting helps from him when I was confused; I have obtained inspirations from him when I was depressed; and I have obtained rigorous comments from him when I was not correct on mathematics. I am anxious to have his comments, because they are my precious stairs, which shape my research from style to content. I would like to express my deepest gratitude to Steve.

I would like to thank my second supervisor and friend Dr. Xuelong Li for his discussions, suggestions, and encouragements. With Xuelong's help, I can do my research here directly and I can become familiar with London with little effort.

I appreciate my external examiner Professor Andrew Blake at Microsoft Research Cambridge Lab and internal examiner Professor Shaogang Gong at Queen Mary for their time reviewing this thesis, their patience for my PhD viva, their constructive comments to refine this thesis, and their encouragements for my work.

I would like to thank my fantastic collaborator and friend Professor Xindong Wu (UVM) for his valuable and interesting suggestions.

I would like to thank my fantastic collaborator and friend Professor Christos Faloutsos (CMU) and Mr. Jimeng Sun (CMU) for an interesting collaboration work on tensor analysis for streaming data.

I would like to thank Professor Mario Figueiredo (Instituto Superior Técnico) for his explanation on the boundary problem in the training procedure of the Gaussian mixture model.

I need to thank the financial support from the Overseas Research Students Award Scheme (ORSAS), the Birkbeck College, and the School of Computer Science and Information Systems. Without them, I have no chance to finish my PhD in UK.

I would also like to thank Mr. Phil Gregg, Mr. Graham Sadler, Mr. Andrew Watkins, and Mr. Phil Docking for their help on computing resources. Without their help, I would not have obtained my experimental results in time. I occupied many computers simultaneously (usually more than five) in the laboratory many times.

I would like to thank Ms. Betty Walters and Ms. Gilda Andreani for their help in administration during my study. I thank all members in the department for their encouragements and suggestions.

I would like to thank my good friends, Dr. Renata Camargo (Birkbeck), Mr. Liangliang Cao (UIUC), Dr. Yuzhong Chen (Tokyo), Dr. Jun Cheng (CUHK), Ms. Victoria Gagova (Birkbeck), Dr. Qi Li (UDEL), Dr. Zhifeng Li (CUHK), Mr. Wei Liu (CUHK), Ms. Ju Luan (CUHK), Mr. Bo Luo (PSU), Mr. Xiaolong Ma (SUNY-SB), Dr. Mingli Song (ZJU), Mr. Jimeng Sun (CMU), Mr. Wei Wang (UC-Davis), Mr. Xiaogang Wang (MIT), Dr. Yonggang Wen (MIT), Dr. Hui Xiong (Rutgers), Dr. Dong Xu (Columbia), Mr. Jin Xu (NLPR), Mr. Tianqiang Yuan (CUHK), Dr. Dell Zhang (Birkbeck), etc. Moreover, I especially need to thank Mr. Xiaolong Ma for his friendly support. I also need to thank Dr. Qi Li for useful discussions.

Finally, I owe my family a lot for many things. I thank all my family members for their extreme understanding and infinite support of my PhD study and academic pursuits.

1. Introduction

Linear subspace methods [58] have been used as an important pre–processing step in many applications of classification for dimension reduction or subspace selection, because of the so called curse of dimensionality [7]. The aim of the linear subspace methods is to project the original high dimensional feature space to a low dimensional subspace. Many methods have been proposed for selecting the low dimensional subspace, e.g., principal component analysis (PCA) [64] and linear discriminant analysis (LDA) [103]. PCA finds a subspace, which minimizes reconstruction error, while LDA finds a subspace, which separates different classes in the projected low dimensional space.

In this thesis, we mainly focus on discriminative linear subspace methods, especially on LDA, because LDA has been widely used in classification tasks in computer vision research, such as image segmentation [97], relevance feedback in content based image retrieval [139][152][153][154], image database indexing [113], video shots classification [43], medical image analysis [116], object recognition and categorization [54], natural scene classification [156], face recognition [158][118][34][168], gait recognition [46][78][79][166][165][142], fingerprint recognition [59], palmprint recognition [62][179], texture classification [133], and hand writing classification [119].

The following view of LDA is taken in this thesis: when samples are drawn from different Gaussian distributions [60] with identical covariance matrices, LDA maximizes the arithmetic mean of the Kullback–Leibler (KL) [21] divergences between all pairs of distributions after projection into a subspace. From this point of view, LDA has the following problems: 1) heteroscedastic problem [29][28][35][70][61][81][99]: LDA ignores the discriminative information, which is present when the covariance matrices change between classes; 2) multimodal problem [51][28]: each class is modelled by a single Gaussian distribution; and 3) class separation problem [103][95][96][99][146]: LDA merges classes which are close together in the original feature space. We call these three problems model based problems, because they are part of the limitations of the definition of LDA.

Apart from the model based problems, LDA also has the small sample size (SSS) problem [38][49][139][123][19][175][55][173][174][180], when the number of training samples is less than the dimension of the feature space. In the LDA model, two scatter matrices are calculated. These are the between class scatter matrix [39] and the within class scatter matrix [39]. The between class scatter matrix measures the separation between different classes. The larger the volume (e.g., the trace value) of the matrix is, the better the different classes are separated. The within class scatter matrix describes the scatter of samples around their respective class centers. The smaller the volume of the matrix is, the closer the samples are to their respective class centers. LDA finds a subspace, which maximizes the ratio between the trace of the projected between class scatter matrix and the trace of the projected within class scatter matrix. The solution is given by an eigenvalue decomposition [44] of the product of the inverse of the within class scatter matrix and the between class scatter matrix. In many real applications, LDA cannot be applied in this straightforward way, because the rank of the within class scatter is deficient [47], i.e., the inverse of the matrix does not exist.

To deal with the model based problems, it is important to develop a flexible framework. Because LDA is equal to the maximization of the arithmetic mean of the KL divergences when samples are obtained from different Gaussian distributions with identical covariances, we develop a general averaged divergences analysis framework, which extends LDA in two ways: 1) generalizing the KL divergences to the Bregman divergence [14], which is a general distortion measure of probability distributions and 2) generalizing the arithmetic mean to the generalized mean [48], which includes as special cases a large number of mean functions, e.g., the arithmetic mean, the geometric mean [20], and the harmonic mean [1][48]. Because this framework takes different covariance matrices into account, it can make better use of the information in heteroscedastic data. Then we combine the Gaussian Mixture Model (GMM) [30] with the framework to reduce the multimodal problem. In applications, LDA tends to merge classes, which are close in the original high dimensional space. This problem is reduced with the proposed general averaged divergences analysis framework by using geometric mean for subspace selection. Three criteria for subspace selection are described: 1) maximization of the geometric mean of the divergences; 2) maximization of the geometric mean of normalized divergences; and 3) maximization of the geometric mean of all divergences (both the divergences and the normalized divergences). Experiments with computer generated data and digitized hand writing [53] show that the combination of the geometric mean based criteria and the KL divergence significantly reduces the heteroscedastic problem, the multimodal problem and the class separation problem.

To deal with the SSS problem, the following two steps are conducted:

1) the LDA criterion is replaced by the difference between the trace of the between class scatter matrix in the projected subspace and the weighted trace of the within class scatter matrix in the projected subspace. The new criterion is named the differential scatter discriminant criterion (DSDC) [143][149]. The relationship between LDA and DSDC is discussed in detail in Chapter 2;

2) In computer vision research, samples are often tensors, i.e., multidimensional arrays. For example, the averaged gait image [45][91], which is the feature used for gait recognition, is a matrix, or a second order tensor; a face image, is also a matrix in face recognition [182]; a color image [54] in object recognition is a third order tensor or a three dimensional array; and a color video shot (a color image sequence) [43] in video retrieval is a forth order tensor or a four dimensional array. Therefore, DSDC is reformulated through operations in multilinear algebra [115][75] or tensor algebra [115][75]. That is we substitute the multilinear algebra operations, e.g., the tensor product, the tensor contraction, and the mode product, for the linear algebra operations, e.g., the matrix product, the matrix transpose, and the trace operation. Then we can directly replace the vectors, which are used to represent vectorized samples, with tensors, which are used to represent the original samples.

The combination of the two steps described above is named the general tensor discriminant analysis (GTDA) [144][147]. By this replacement, the SSS problem can be significantly reduced, because we need not to estimate a large projection matrix at a time. That is we estimate a series of small projection matrices iteratively by using the alternating projection method, which obtains each small projection matrix with all the other fixed projection matrices in an iterative way, i.e., an alternating projection method for optimization decouples a projection matrix from the others. In each iteration, the number of unknown parameters (the size of a projection matrix) in GTDA is much less than that of in LDA.

GTDA is motivated by the successes of the tensor rank one analysis (TR1A) [132], general tensor analysis (GTA) [75][162][171], and the two dimensional LDA (2DLDA) [174] for face recognition. The benefits of GTDA are: 1) reduction of the SSS problem for subsequent classification, e.g., by LDA; 2) preservation of discriminative information in training tensors, while PCA, TR1A, and GTA do not guarantee this; 3) provision with stable recognition rates because the optimization algorithm of GTDA converges, while that of 2DLDA does not; and 4) acceptance of the general tensors as input, while 2DLDA only accepts matrices as input.

We then apply the proposed GTDA to appearance [82] based human gait recognition [45][46][91][92]. For appearance based gait recognition, the averaged gait images are suited for human gait recognition, because: 1) the averaged gait images of the same person share similar visual effects under different circumstances; and 2) the averaged gait images of different people even under the same circumstance are very different. Motivated by the successes of Gabor function [32][80] based image decompositions for image understanding and object recognition, we develop three different Gabor function based image representations [144]: 1) the sum of Gabor functions over directions based representation (GaborD), 2) the sum of Gabor functions over scales based representation (GaborS), and 3) the sum of Gabor functions over scales and directions based representation (GaborSD). These representations are applied to recognize people from their averaged gait images. A large number of experiments were carried out to evaluate the effectiveness (recognition rate) of gait recognition based on the following three successive steps: 1) the Gabor/GaborD/GaborS/ GaborSD image representations, 2) GTDA to extract features from the Gabor/ GaborD/GaborS/GaborSD image representations, and 3) applying LDA for recognition. The proposed methods achieve sound performance for gait recognition based on the USF HumanID Database [126]. Experimental comparisons are made with nine state of the art classification methods [46][66][126][167][173] in gait recognition.

Finally, vector based learning[1] is extended to accept tensors as input. This results in the supervised tensor learning (STL) framework [149][150], which is the

[1] Vector based learning means the traditional classification technique, which accepts vectors as input. In vector based learning, a projection vector $\vec{w} \in R^L$ and a bias $b \in R$ are learnt to

multilinear extension of the convex optimization [11] based learning. To obtain the solution of an STL based learning algorithm, an alternating projection method is designed. Based on STL and its alternating projection optimization algorithm, we illustrate some examples. That is we extend the soft margin support vector machine (SVM) [161][15], the nu–SVM [130][128], the least squares SVM [137][138], the minimax probability machine (MPM) [74][135], the Fisher discriminant analysis (FDA) [37][30][69], the distance metric learning (DML) [169] to their tensor versions, which are the soft margin support tensor machine (STM), the nu–STM, the least squares STM, the tensor MPM (TMPM) [150], the tensor FDA (TFDA), and the multiple distance metrices learning (MDML), respectively. With STL, we also introduce a method for feature extraction through an iterative way [132] and develop the tensor rank one discriminant analysis (TR1DA) [145][143] as an example. The experiments for image classification demonstrate TMPM reduces the overfitting problem in MPM. The experiments for the elapsed time problem in human gait recognition show TR1DA is more effective than PCA, LDA, and TR1A.

---

determine the class label of a sample $\vec{x} \in R^L$ according to a linear decision function $y(\vec{x}) = \text{sign}\left[\vec{w}^T \vec{x} + b\right]$. The $\vec{w}$ and b are obtained based on a learning model, e.g., minimax probability machine (MPM), which is based on N training samples associated with labels $\left\{\vec{x}_i \in R^L, y_i\right\}$, where y_i is the class label, $y_i \in \{+1, -1\}$, and $1 \le i \le N$.

Thesis Organization

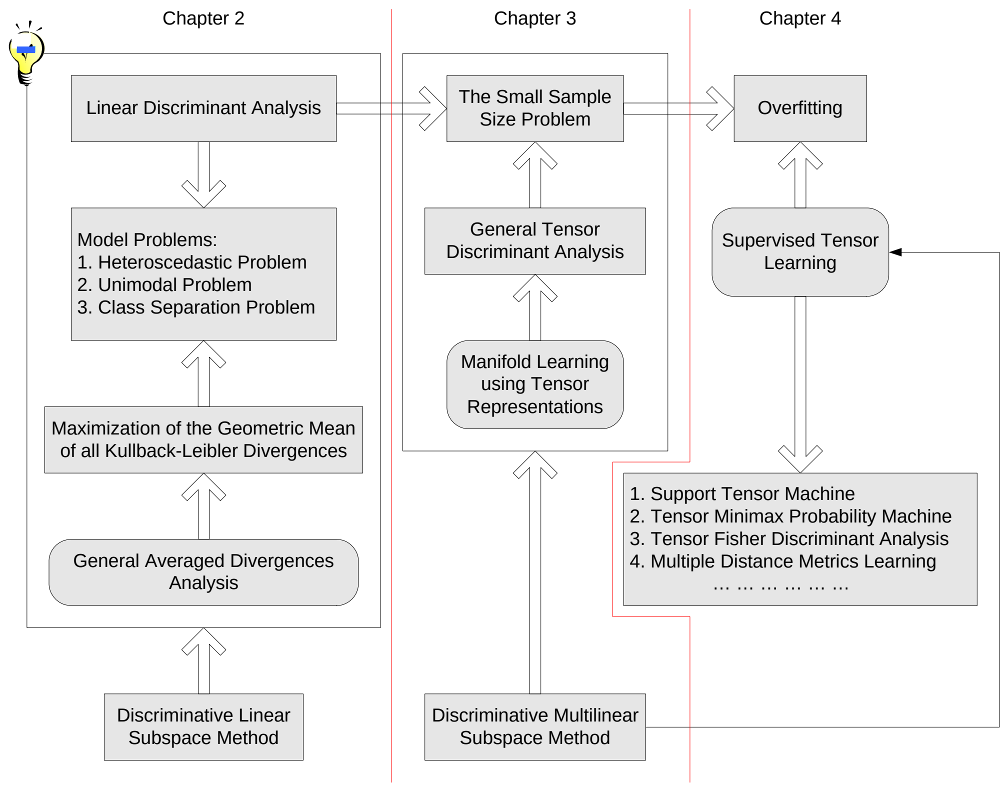

Chapter 2
Chapter 3
Chapter 4
Linear Discriminant Analysis
The Small Sample Size Problem
Overfitting
Model Problems:
1. Heteroscedastic Problem
2. Unimodal Problem
3. Class Separation Problem
General Tensor Discriminant Analysis
Supervised Tensor Learning
Manifold Learning using Tensor Representations
Maximization of the Geometric Mean of all Kullback-Leibler Divergences
General Averaged Divergences Analysis
1. Support Tensor Machine
2. Tensor Minimax Probability Machine
3. Tensor Fisher Discriminant Analysis
4. Multiple Distance Metrics Learning
… … … … … …
Discriminative Linear Subspace Method
Discriminative Multilinear Subspace Method

Figure 1.1. The plan of the thesis.

LDA selects subspace to separate different classes in the projected low dimensional subspace and it has been widely applied for classification tasks in computer vision research. However, LDA has two types of problems: 1) the model based problems, which are led by its definition and 2) the small sample size (SSS) problem, when the number of training samples is less than the dimension of the feature space. To deal with the model based problems, we develop the maximization of the geometric mean of all Kullback-Leibler divergences under the general averaged divergences analysis framework in Chapter 2, as shown in Figure 1.1. To deal with the SSS problem, we propose a general tensor discriminant analysis (GTDA) for subspace selection based on tensor algebra. GTDA is also extended for popular manifold learning algorithms in Chapter 3, as shown in Figure 1.1. The SSS problem is relevant to the overfitting problem in vector based learning algorithms for classification. Both problems arise when the number of training samples is small. In Chapter 4, we apply tensor algebra to extend vector based learning algorithms to accept tensors as input to reduce the overfitting problem. A number of examples are provided in this Chapter, as shown in Figure 1.1.

In Chapter 2, we review two important linear subspace methods [58], namely principal component analysis (PCA) [64] and linear discriminant analysis (LDA) [103]. We then analyze the problems of LDA and review some extensions [19][28][29][38][49][51][55][61][70][95][96][99][103][123][139][173][174][175][180] of LDA to reduce these problems. We also give a new point of view [146] on discriminative subspace selection and develop a new framework for subspace selection, the general averaged divergences analysis [146]. This framework allows a range of different criteria for assessing subspaces. Based on the new framework, we investigate geometric mean [20] for discriminative subspace selection and develop a method for the maximization of the geometric mean of all Kullback–Leibler (KL) [21] divergences (MGMKLD) [146] by combining the maximization of the geometric mean of all divergences and the KL divergence for subspace selection. A large number of experiments are conducted to demonstrate the effectiveness of the new discriminative subspace selection method compared with LDA and its representative extensions.

In Chapter 3, we focus on multilinear subspace methods, because the objects in computer vision research are often tensors [75]. Firstly, tensor algebra [115][75] is briefly introduced. It is the mathematical fundamental material of this Chapter. After that, unsupervised learning techniques, such as TR1A [132] and GTA [75][162][171], are reviewed. We also give a brief introduction of 2LDA [174]. Motivated by the success of 2DLDA in face recognition, we then develop GTDA, which includes the following parts: 1) the LDA criterion is replaced by DSDC [143][149]. The relationship between LDA and DSDC is discussed in detail; and 2) DSDC is reformulated through operations in multilinear algebra [115][75]. Based on the reformulated DSDC, vectors can be replaced with tensors, i.e., retaining the original format of samples; 3) an alternating projection optimization procedure is developed to obtain the solution of GTDA; 4) provide the mathematical proof of the convergence of the alternating projection optimization procedure for calculating the projection matrices; and 5) the computational complexity is analyzed. Finally, the proposed GTDA combined with LDA and the nearest neighbour classifier is utilized for appearance based human gait recognition [45][46][91][92]. Compared with previous algorithms, the newly presented algorithms achieve better recognition rates.

In Chapter 4, a supervised tensor learning (STL) framework [149][150] is developed based on the similar idea to reduce the SSS problem in Chapter 3. We first introduce convex optimization [11] and convex optimization based learning. Then we propose the STL framework associated with the alternating projection method for optimization. Based on STL and its alternating projection optimization algorithm, we generalize the support vector machine (SVM) [161][15][130][128] [137][138], the minimax probability machine (MPM) [74][135], the Fisher discriminant analysis (FDA) [37][30] [69], and the distance metric learning (DML) [169], as the support tensor machine, the tensor minimax probability machine, the tensor Fisher discriminant analysis, and the multiple distance metrics learning, respectively. We also propose an iterative feature extraction method based on STL. As an example, we develop the tensor rank one discriminant analysis (TR1DA). Experiments are conducted based on the tensor minimax probability machine and TR1DA.

Chapter 5 concludes.

The main contributions of the thesis are three folds:

1. Develop a discriminative subspace selection framework, i.e., general averaged divergences analysis. Based on this framework, a special case, i.e., maximization of the geometric mean of all Kullback–Leibler (KL) divergences, is given to significantly reduce the class separation problem raised by imbalanced distributions of KL divergences between different classes. Moreover, it is also compatible with the heteroscedastic property of data and deals with samples drawn from mixture of Gaussians naturally. Empirical studies demonstrate that it outperforms LDA and its representative extensions;
2. Develop the general tensor discriminant analysis (GTDA) to reduce the small sample size (SSS) problem. Unlike all existing tensor based discriminative subspace selection algorithms, GTDA converges in the training stage. Moreover, a full mathematical proof is given. To our best knowledge, this is the first work in the world to give both a converged algorithm and a mathematical proof. Again, this proof can be also applied to justify whether a tensor based algorithm converges or not by checking

the convexity of its objective function. By applying GTDA to human gait recognition, we achieve the state-of-the-art recognition accuracy; and

3. By applying tensor algebra to vector based learning, we finally develop a supervised tensor learning framework. The significance of the framework is we can conveniently generalize different vector based classifiers to tensor based classifiers to reduce the over–fitting problem. For example, we generalize the support vector machine to the support tensor machine and give out the error bound; we generalize the Fisher discriminant analysis and the minimax probability machine to the tensor Fisher discriminant analysis and the tensor minimax probability machine, respectively, to overcome the matrix singular problem; and we generalize distance metric learning to multiple distance metrics learning to make it computable for appearance based recognition tasks. Finally, an iterative feature extraction model is given based on supervised tensor learning. Empirical studies show the power of supervised tensor learning and the iterative feature extraction model.

Publications

The research of this thesis has resulted in the following research papers:

1. D. Tao, X. Li, X. Wu, and S. J. Maybank, "General Tensor Discriminant Analysis and Gabor Features for Gait Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence* (**TPAMI**), 2007. [Chapter 3]
2. D. Tao, X. Li, X. Wu, and S. J. Maybank, "Tensor Rank One Discriminant Analysis," Submitted to *IEEE Transactions on Pattern Analysis and Machine Intelligence* (**TPAMI**). (Under Major Revision) [Chapter 4]
3. D. Tao, X. Li, X. Wu, and S. J. Maybank, "General Averaged Divergences Analysis," Submitted to *IEEE Transactions on Pattern Analysis and Machine Intelligence* (**TPAMI**). (Under Major Revision) [Chapter 2]
4. D. Tao, X. Li, and S. J. Maybank, "Negative Samples Analysis in Relevance Feedback," *IEEE Transactions on Knowledge and Data Engineering* (**TKDE**), 2007.
5. D. Tao, X. Li, W. Hu, S. J. Maybank, and X. Wu, "Supervised Tensor Learning: A Framework," *Knowledge and Information Systems* (Springer), 2007. [Chapter 4]
6. D. Tao, X. Li, X. Wu, and S. J. Maybank, "Human Carrying Status in Visual Surveillance," *IEEE Int'l Conf. on Computer Vision and Pattern Recognition* (**CVPR**), pp. 1,670–1,677, 2006. (Acceptance rate: ~20%) [Chapter 3]
7. D. Tao, X. Li, X. Wu, and S. J. Maybank, "Elapsed Time in Human Gait Recognition: A New Approach," *IEEE Int'l Conf. on Acoustics, Speech, and Signal Processing* (ICASSP), vol. 2, pp. 177–180, 2006. [Chapter 4]
8. D. Tao, X. Li, W. Hu, S. J. Maybank, and X. Wu, "Supervised Tensor Learning," *IEEE Int'l Conf. on Data Mining* (**ICDM**), pp. 450–457, 2005. (Acceptance rate: ~11%) [Chapter 4]
9. D. Tao, X. Li, W. Hu, and S. J. Maybank, "Stable Third–Order Tensor Representation for Color Image Classification," *IEEE Int'l Conf. on Web Intelligence* (WI), pp. 641–644, 2005.

2. Discriminative Linear Subspace Methods

The linear subspace method (LSM) [2][21][50][109][107] has been developed and demonstrated to be a powerful tool in pattern recognition and computer vision research fields. LSM finds a matrix $U \in R^{L \times L'}$ to transform the high dimensional sample $\vec{x} \in R^{L}$ to a low dimensional sample $\vec{y} \in R^{L'}$, i.e., $\vec{y} = U^{T}\vec{x}$. There are two major categories of LSM algorithms, which are focused on either feature selection or dimension reduction, respectively. A feature selection algorithm selects a (very small) number of most effective features from the entire feature pool. That is the un–selected features are not utilized. In feature selection, the linear transformation matrix U has the following properties: 1) the entries of U are 1 or 0; 2) the inner product of any two columns of U is 0; and 3) the sum of all entries of any column of U is 1. A dimension reduction algorithm finds several sets of coefficients, and with each set of coefficients the original features are weighted and summed to produce a new feature. By this means, several (less than the number of the original features) new low dimensional samples are "generated" to preserve as much as possible the information (e.g., reconstructive information or discriminative information) carried by the original high dimensional samples. In this thesis, we focus on algorithms for dimension reduction.

From the viewpoint of modelling, LSM can be used with a large number of models varying from reconstructive models to discriminative models. A reconstructive LSM minimizes $\sum_{i=1}^{N} L\left(\left\|\vec{x}_i - U\vec{y}_i\right\|_H\right)$, where we have N training samples $\vec{x}_i$ on hand; the projection matrix is U; $\vec{y}_i$ is the low dimensional representation of $\vec{x}_i$; $\|\cdot\|_H$ is a norm; and $L(\cdot)$ is a loss function [161][128]. Principal component analysis (PCA) [64] is an example of a reconstructive model. On the other side, discriminative models, e.g., linear discriminant analysis (LDA) [39], are utilized for classification. A discriminative LSM maximizes an objective function to separate different classes in the projected low dimensional subspace. Both reconstructive and discriminative models are widely used in many real–world applications, such as biometrics [182][68], bioinformatics [31], and multimedia information management [24][139].

In this Chapter, we mainly focus on the discriminative LSM, especially LDA, because it is the most popular algorithm in dimension reduction (or subspace selection) for classification. If samples are sampled from Gaussian distributions with identical covariance matrices, LDA maximizes the arithmetic mean value of the Kullback–Leibler (KL) [21] divergences between different classes. Based on this point of view, it is not difficult to see that LDA has the following problems:

1) Heteroscedastic problem [29][28][70][61][99]: LDA models different classes with identical covariance matrices. Therefore, it fails to take account of any variations in the covariance matrices between different classes;

2) Multimodal problem [51][28]: In many applications, samples in each class can not be approximated by a single Gaussian. Instead, a Gaussian mixture model (GMM) [39][30] is required. However, LDA models each class by a single Gaussian distribution;

3) Class separation problem [103][95][96][99][146]: In applications, distances between different classes are different and LDA tends to merge classes which are close together in the original feature space.

The first two problems have been well studied in the past few years and a number of extensions of LDA have been generated to deal with them. Although some methods [103][95][96][99] have been proposed to reduce the third problem, it is still not well solved. In this Chapter, to further reduce the class separation problem, we first generalize LDA to obtain a general averaged divergences analysis, which extends LDA from two aspects: 1) the KL divergence is replaced by the Bregman divergence [14]; and 2) the arithmetic mean is replaced by the generalized mean function [48]. By choosing different options in 1) and 2), a series of subspace selection algorithms are obtained, with LDA included in as a special case.

Under the general averaged divergences analysis, we investigate the effectiveness of geometric mean [20] based subspace selection in solving the class separation problem. The geometric mean amplifies the effects of small divergences and at the same time reduces the effects of large divergences. Next, the maximization of the geometric mean of the normalized divergences is studied. This turns out not to be suitable for subspace selection, because there exist projection matrices which make all divergences very small and at the same time make all normalized divergences similar in value. We therefore propose a third criterion, maximization of the geometric mean of all divergences (both the divergences and the normalized divergences) or briefly MGMD. It is a combination of the first two.

With MGMD, it is possible to develop different subspace selection methods by choosing different divergences. In this chapter, we select the KL divergence and assume that the samples in each class are obtained by sampling Gaussian distributions. This results in the maximization of a function of all KL divergences (MGMKLD). The name MGMKLD is chosen because the function is closely related to the geometric mean of divergences. We extend MGMKLD to the case in which the samples in each class are sampled from a Gaussian Mixture Model [30]. This gives the multimodal extension of MGMKLD, or M–MGMKLD. Finally, we kernelize [104][109][128][129] MGMKLD to the kernel MGMKLD or briefly KMGMKLD. Preliminary experiments based on synthetic data and handwriting digital data [53] show that MGMKLD achieves much better classification rates than LDA and its several representative extensions taken from the literature.

The Chapter is organized as follows. In §53 and §53, PCA with its kernel extension and LDA with its representative extensions are briefly reviewed, respectively. In §53, the general averaged divergences analysis is proposed. In §53, the geometric mean for subspace selection is investigated. The KL divergence based geometric mean subspace selection is developed in §53. synthetic data based experiments, statistical experiments, and hand writing recognition for justifying the effectiveness of linear subspace methods are given in §53, §53, and §53, respectively. Finally, summary of this Chapter is given in §53. Moreover, all proofs and deductions in this Chapter are given in §53.

Principal Component Analysis

Although PCA [64] is a reconstructive model, it has been successfully applied for classification tasks in computer vision. It extracts the principal eigenspace associated with a set of training samples $\vec{x}_i \in R^L$ ($1 \le i \le n$). Let $S = (1/n)\sum_{i=1}^{n}(\vec{x}_i - \vec{m})(\vec{x}_i - \vec{m})^T$ be the covariance matrix, alternatively called the total–class scatter matrix, of all training samples $\vec{x}_i$, where $\vec{m} = (1/n)\sum_{i=1}^{n}\vec{x}_i$. One solves the eigenvalue equation $\lambda \vec{u}_i = S\vec{u}_i$ for eigenvalues $\lambda_i \ge 0$. The projection matrix $U*$ is spanned by the first L' eigenvectors with the largest eigenvalues, $U* = \left[\vec{u}_i^* |_{i=1}^{L'}\right]$. If $\vec{x}$ is a new sample, then it is projected to $\vec{y} = (U*)^T(\vec{x} - \vec{m})$. The vector $\vec{y}$ is used in place of $\vec{x}$ for representation and classification.

PCA has the following properties. In the following description, we assume the training samples $\vec{x}_i$ are centralized, i.e., $\vec{m} = 0$.

Property 2.1: PCA maximizes the variance in the projected subspace for a given dimension L', i.e.,

$$\arg\max_U \operatorname{tr}\left(U^T S U\right) = \arg\max_U \frac{1}{n}\sum_{i=1}^{n}\left\|U^T \vec{x}_i\right\|_{Fro}^2 . \qquad (2.01)$$

where $\|\cdot\|_{Fro}$ is the Frobenius norm.

Proof: See Appendix.

Property 2.2: The principal eigenspace U in PCA diagonalizes the covariance matrix of the training samples.

Property 2.3: PCA minimizes the reconstruction error, i.e.,

$$\arg\max_U \operatorname{tr}\left(U^T S U\right) = \arg\min_U \frac{1}{n}\sum_{i=1}^{n}\left\|\vec{x}_i - UU^T \vec{x}_i\right\|_{Fro}^2 . \qquad (2.02)$$

Proof: See Appendix.

Property 2.4: PCA decorrelates the training samples in the projected subspace.

Proof: See Appendix.

Property 2.5: PCA maximizes the mutual information between $\vec{x}$ and $\vec{y}$ on Gaussian data.

Proof: See Appendix.

We now study the nonlinear extension of PCA, or the kernel PCA (KPCA) [129], which takes high order stastistics of the training samples into account. Consider a nonlinear mapping:

$$\phi : R^L \to R^H, \quad \vec{x} \mapsto \phi(\vec{x}), \tag{2.03}$$

where $H = n$. Then, in R^H, the covariance matrix is

$$S_\phi = \frac{1}{n}\sum_{i=1}^{n}\left(\phi(\vec{x}_i) - \vec{m}_\phi\right)\left(\phi(\vec{x}_i) - \vec{m}_\phi\right)^T, \tag{2.04}$$

where $\vec{m}_\phi = (1/n)\sum_{i=1}^{n}\phi(\vec{x}_i)$ is the mean vector of all training samples in R^H. The $\phi(\vec{x}_i)\phi^T(\vec{x}_i)$ is a linear operator on the range of ϕ in R^H. Suggested by Schölkopf et al. [128][129], the mapping is defined as $\vec{x} \mapsto \phi(\vec{x}_i)\langle\phi(\vec{x}_i), \vec{x}\rangle$. The next step in KPCA is to find the eigenvalue decomposition on S_ϕ.

$$\lambda\vec{v} = S_\phi\vec{v}. \tag{2.05}$$

Because all solutions $\vec{v}$ with $\lambda \neq 0$ are the linear combinations of $\phi(\vec{x}_i)$, $1 \le i \le n$, we have

$$\lambda\langle\phi(\vec{x}_i), \vec{v}\rangle = \langle\phi(\vec{x}_i), S_\phi\vec{v}\rangle, \text{ for all } 1 \le i \le n. \tag{2.06}$$

Replace $\vec{v}$ with $\sum_{i=1}^{n}\alpha_i\phi(\vec{x}_i)$ in (2.06), we have

$$\begin{aligned} &\lambda\sum_{j=1}^{n}\alpha_j\left\langle\phi(\vec{x}_i), \phi(\vec{x}_j)\right\rangle \\ &= \sum_{j=1}^{n}\alpha_j\left\langle \phi(\vec{x}_i), \frac{1}{n}\sum_{k=1}^{n} \begin{matrix}\left(\phi(\vec{x}_k) - \frac{1}{n}\sum_{k=1}^{n}\phi(\vec{x}_k)\right) \\ \left\langle\left(\phi(\vec{x}_k) - \frac{1}{n}\sum_{k=1}^{n}\phi(\vec{x}_k)\right), \phi(\vec{x}_j)\right\rangle\end{matrix} \right\rangle \end{aligned} \tag{2.07}$$

for all $1 \le i \le n$.

In terms of the $n \times n$ kernel Gram matrix [128][129] $K_{ij} := \left\langle\phi(\vec{x}_i), \phi(\vec{x}_j)\right\rangle = k(\vec{x}_i, \vec{x}_j)$, (2.07) is simplified as

$$\lambda_i K\vec{\alpha}_i = K(I - M)(I - M)^T K\vec{\alpha}_i \tag{2.08}$$

where $\vec{\alpha}_i$ is a column vector and it is the eigenvectors of $(I - M)(I - M)^T K$; λ_i is the eigenvalues of K; $k(\vec{x}_i, \vec{z}) = \langle\phi(\vec{x}_i), \phi(\vec{z})\rangle$ is the kernel function [128][129]; and all entries in $M = \left[m_{ij}\right]_{1\le i;\, j\le n} \in R^{n\times n}$ are $1/n$. The projection matrix Λ in R^H is spanned by the first $H'(H' < H = n)$ $\vec{\alpha}_i$ with the largest eigenvalues, i.e., $\Lambda = [\vec{\alpha}_i]_{1 \le i \le n}$.

After obtaining the linear combination coefficients, we can project a given sample $\vec{z}$ to the subspace constructed by KPCA according to

$$\begin{aligned} (X_\phi \Lambda)^T \phi(\vec{z}) &= \Lambda^T X_\phi^T \phi(\vec{z}) \\ &= \Lambda^T \left[k(\vec{x}_1, \vec{z}), k(\vec{x}_2, \vec{z}), \cdots, k(\vec{x}_n, \vec{z}) \right], \end{aligned} \tag{2.09}$$

where $X_\phi = [\phi(\vec{x}_i)]_{1 \le i \le n}$.

Linear Discriminant Analysis

LDA [103] finds in the feature space a low dimensional subspace where the different classes of samples remain well separated after projection to this subspace. The subspace is spanned by a set of vectors, which are denoted as $U = [\vec{u}_1, \ldots, \vec{u}_{L'}]$. It is assumed that a training set of samples is available. The training set is divided into c classes. The ith class contains n_i samples $\vec{x}_{i;j}$ $(1 \le j \le n_i)$, and has a mean value $\vec{m}_i = (1/n_i)\sum_{j=1}^{n_i} \vec{x}_{i;j}$. The between class scatter matrix S_b and the within class scatter matrix S_w are defined by

$$\begin{cases} S_b = \dfrac{1}{n}\displaystyle\sum_{i=1}^{c} n_i (\vec{m}_i - \vec{m})(\vec{m}_i - \vec{m})^T \\ S_w = \dfrac{1}{n}\displaystyle\sum_{i=1}^{c}\sum_{j=1}^{n_i} (\vec{x}_{i;j} - \vec{m}_i)(\vec{x}_{i;j} - \vec{m}_i)^T \end{cases} \tag{2.10}$$

where c is the number of classes; $n = \sum_{i=1}^{c} n_i$ is the size of the training set; and $\vec{m} = (1/n)\sum_{i=1}^{c}\sum_{j=1}^{n_i} \vec{x}_{i;j}$ is the mean vector of all training samples. Meanwhile, $S_t = (1/n)\sum_{i=1}^{c}\sum_{j=1}^{n_i} (\vec{x}_{i;j} - \vec{m})(\vec{x}_{i;j} - \vec{m})^T = S_b + S_w$ is the covariance matrix of all samples.

The projection matrix U^* of LDA is chosen to maximize the ratio between S_b and S_w in the projected subspace, i.e.,

$$U^* = \arg\max_{U} \operatorname{tr}\left((U^T S_w U)^{-1} (U^T S_b U) \right). \tag{2.11}$$

The projection matrix U^* is computed from the eigenvectors of $S_w^{-1} S_b$, under the assumption that S_w is invertible. If c equals to 2, LDA reduces to the Fisher discriminant analysis [37]; otherwise LDA is known as the Rao discriminant analysis [122]. Because $\operatorname{rank}(S_b) \le c - 1$, we have $L' \le c - 1$, i.e., the maximum dimension of the projected subspace for LDA is $\min(c-1, L-1)$.

If separate classes are sampled from Gaussian distributions, all with identical covariance matrices, then LDA maximizes the mean value of the KL divergences between different classes. This result will be proved in §122.

LDA encounters the following problems, which are:

1) Heteroscedastic problem [29][28][70][61][99]: LDA discards the discriminative information preserved in covariance matrices of different classes;
2) Multimodal problem [51][28]: LDA models each class by a single Gaussian distribution, so it cannot find a suitable projection for classification when samples are sampled from complex distributions, e.g., GMM;
3) Class separation problem [103][95][96][99][146]: LDA tends to merge classes which are close together in the original feature space.
Furthermore, when the size of the training set is smaller than the dimension of the feature space, LDA has the small sample size (SSS) problem [19][30][38][39] [49][55][123][139][173][174][175][180].
In the following, we review some representative solutions for these problems. Furthermore, we mention some alternatives of LDA [39], namely the nonparametric discriminant analysis [39][40], and the kernel extension of LDA [104][109] [128][129].

Heteroscedastic Problem

LDA does not fully utilize the discriminative information contained in the covariances of different classes. As a result, it cannot find a suitable projection direction when different classes share the same mean, as shown in Figure 2.1.
In the past decades, a large number of extensions based on LDA were developed to reduce this problem. For example,

- Decell and Mayekar [29] proposed a method to obtain a subspace to maximize the average interclass divergence, which measures the separations between the classes. This criterion takes into account the discriminative information preserved in the covariances of different classes. The projection matrix U^* is calculated by maximizing,

$$J_D = \operatorname{tr}\left[\sum_{i=1}^{c}\left(U^T S_i U\right)^{-1}\left(U^T \sum_{j=1;\, j\neq i}^{c}\left(S_j + \left(\vec{m}_j - \vec{m}_i\right)\left(\vec{m}_j - \vec{m}_i\right)^T\right)U\right)\right] - c\left(c-1\right)L', \tag{2.12}$$

where S_i is the i^{th} class covariance matrix ($1 \le i \le c$); $\vec{m}_i$ is the mean vector of the samples in the i^{th} class ($1 \le i \le c$); c is the number of classes of the training set; L' is the number of selected features; and U is the projection matrix to obtain the low dimensional representation.

De la Torre and Kanade [28] developed the oriented discriminant analysis (ODA) based on the objective function defined in (2.12), but used iterative majorization to obtain a solution. The iterative majorization speeds up the training stage.

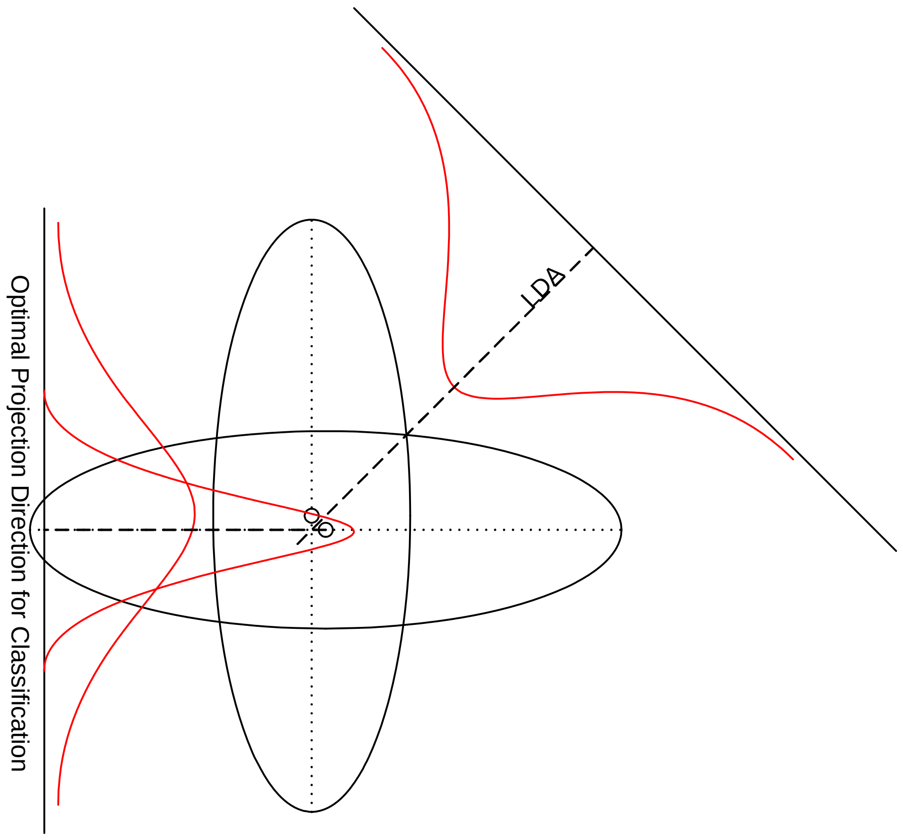


Figure 2.1. LDA fails to find the optimal projection direction for classification, because it does not utilize the discriminative information preserved in the class covariances.

- Campbell [17] has shown LDA is related to the maximum likelihood estimation of parameters for a Gaussian model based on the following two assumptions: 1) all class discriminative information resides in a low dimensional subspace of the original high dimensiaon feature space and 2) the within class covariances are identical for all classes. Kumar and Andreou [70] developed the heteroscedastic discriminant analysis (HDA) by dropping the identical class covariances assumption. The projection matrix U^* is calculated by maximizing,

$$J_K = n\log\left|U_{L-L'}^T S U_{L-L'}\right| + \sum_{i=1}^{c} n_i \log\left|U_{L'}^T S_i U_{L'}\right| - 2n\log\left|U\right|, \tag{2.13}$$

where $U = \left[U_{L'} \quad U_{L-L'}\right]$ is the full transformation matrix; $U_{L'}$ is the transformation submatrix to select the discriminative subspace; $\left|\cdot\right| \square \det\left(\cdot\right)$; S_i is the i^{th} class covariance matrix ($1 \le i \le c$); S is the covariance matrix of all training samples; c is the number of classes of the training set; and L' is the number of selected features. Furthermore, the projection matrix U^* is obtained by maximizing J_K through the gradient steepest ascent algorithm.

- Jelinek [61] proposed a different way to deal with the heteroscedastic problem in subspace selection by the gradient steepest ascent method to find the projection matrix U^* by maximizing,

$$J_J = n\log\left|U^T S_b U\right| - \sum_{i=1}^{c} n_i \log\left|U^T S_i U\right|, \tag{2.14}$$

where $\left|\cdot\right| \square \det\left(\cdot\right)$; S_i is the covariance matrix of the i^{th} class; S_b is the between class scatter matrix defined in (2.10); n_i is the number of samples in the i^{th} class ($1 \le i \le c$); c is the number of classes of the training set; and U is the projection matrix to obtain the low dimensional representation.

- Loog and Duin [99] introduced the Chernoff criterion to heteroscedasticize LDA, i.e., the heteroscedastic extension of LDA (HLDA). The projection matrix U^* in HLDA is obtained by maximizing,

$$\begin{aligned} J_L = \sum_{i=1}^{c-1}\sum_{j=i+1}^{c} q_i q_j S_w^{-1} \times S_w^{1/2} \Big(& \left(S_w^{-1/2} S_{ij} S_w^{1/2}\right)^{-1/2} \times S_w^{-1/2}\left(\vec{m}_i - \vec{m}_j\right)\left(\vec{m}_i - \vec{m}_j\right)^T \\ & \times S_w^{-1/2}\left(S_w^{-1/2} S_{ij} S_w^{1/2}\right)^{-1/2} + \frac{1}{\pi_i \pi_j}\Big(\log\left(S_w^{-1/2} S_{ij} S_w^{1/2}\right) \\ & - \pi_i \log\left(S_w^{-1/2} S_i S_w^{1/2}\right) - \pi_j \log\left(S_w^{-1/2} S_j S_w^{1/2}\right)\Big)\Big) S_w^{1/2}, \end{aligned} \tag{2.15}$$

where $q_i = n_i \big/ \sum_{k=1}^{c} n_k$ is the prior probability of the i^{th} class; S_i is the covariance matrix of the i^{th} class; $\pi_i = q_i \big/ \left(q_i + q_j\right)$; $\pi_j = q_j \big/ \left(q_i + q_j\right)$; $S_{ij} = \pi_i S_i + \pi_j S_j$; S_w is the within class scatter matrix defined in (2.10);

n_i is the number of samples in the ith class ($1 \le i \le c$); and c is the number of classes of the training set.

The projection matrix U^* of HLDA is constructed by the eigenvectors of the J_L corresponding to the largest eigenvalues.

Multimodal problem

The direct way to deal with the multimodal problem is to model each class by a GMM [39][30]. Two representative works are as following,

- Hastie and Tibshirani [51] combined GMM with LDA based on the fact that LDA is equivalent to maximum likelihood classification when each class is modelled by a single Gaussian distribution. The extension directly replaces the original single Gaussian in each class by a Gaussian mixture model in Campbell's result [17], as shown in (2.13).
- De la Torre and Kanade [28] generalized ODA defined in (2.12) for the multimodal case as the multimodal ODA (MODA) by combining it with GMM learnt by the normalized cut [131][177]. Each class is modelled by a GMM. The aim of MODA is to find a projection matrix U^* to maximize

$$J_{MODA} = \sum_{i=1}^{c} \sum_{\substack{j=1 \\ j \neq i}}^{c} \sum_{k=1}^{c_i} \sum_{l=1}^{c_j} \mathrm{tr} \begin{pmatrix} \left(U^T S_{i;k} U\right)^{-1} U^T \\ \times \left(\left(\vec{m}_{i;k} - \vec{m}_{j;l}\right)\left(\vec{m}_{i;k} - \vec{m}_{j;l}\right)^T + S_{j;l} \right) U \end{pmatrix}, \qquad (2.16)$$

where c_i is the number of subclusters of the ith class; $S_{i;k}$ is the covariance matrix of the kth subcluster of the ith class; $\vec{m}_{i;k}$ is the mean vector of the kth subcluster of the ith class; c is the number of classes of the training set; and U is the projection matrix to obtain the low dimensional representation.

Class separation problem

One of the most severe problems in LDA is the class separation problem, i.e., LDA merges classes which are close together in the original feature space. As pointed out by McLachlan in [103], Lotlikar and Kothari in [95], Loog et al. in [98], and Lu et al. in [100], this merging of classes significantly reduces the recognition rate. The example in Figure 2.2 shows that LDA is not always optimal for pattern classification. To improve its performance, Lotlikar and Kothari in [95] developed the fractional–step LDA (FS–LDA) by introducing a weighting function. Loog et al. in [98] developed another weighting method for LDA, namely the approximate pairwise accuracy criterion (aPAC). The advantage of aPAC is that the projection matrix can be obtained by the eigenvalue decomposition. Lu et al. in [100] combined the FS–LDA and the direct LDA [175] for very high dimensional problems, such as face recognition [182]. However, both FS–LDA and aPAC do not use the discriminative information in different class covariances. Therefore, when samples are drawn from Gaussians with different covariances, these two methods fail to detect the suitable subspace for classification (their performance could be even worse than LDA). The detailed procedures for FS–LDA and aPAC are as follows,

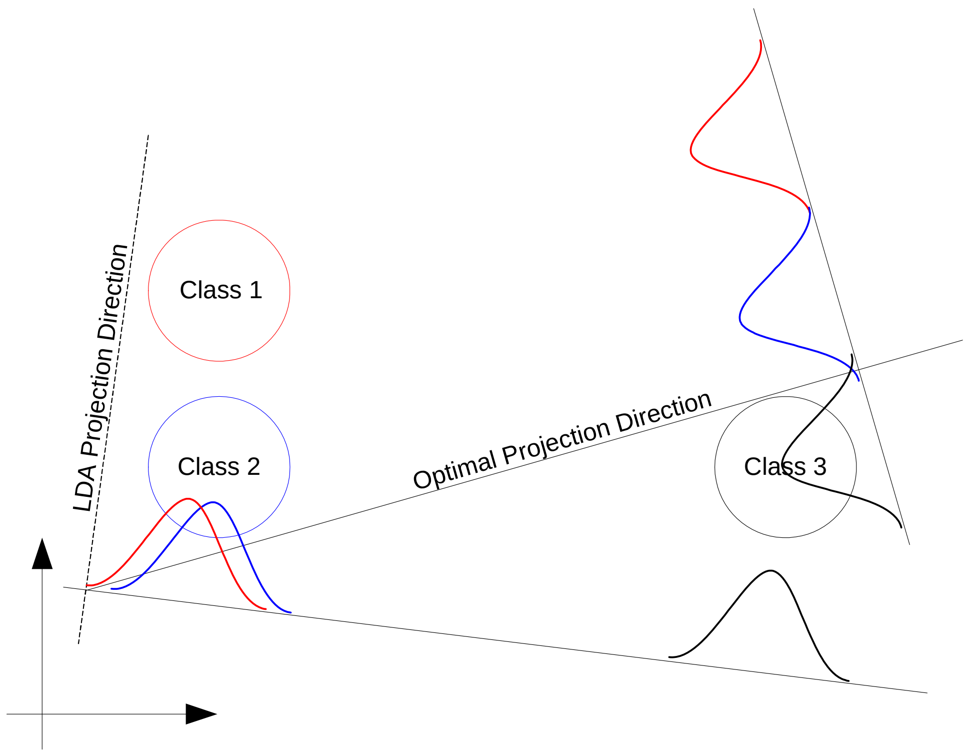


Figure 2.2. The samples in each class are drawn from a Gaussian distribution. LDA finds a projection direction, which merges class 1 and class 2. One of the reasonable projection directions for classification trades the distance between the class 1 and the class 2 against the distance between the class 1, 2 and class 3. This example is a sketch of the synthetic data used in Figure 2.7 in §0.

- Lotlikar and Kothari [95] developed FS–LDA to reduce the *class separation* problem. They found this problem is invoked by non–uniform distances between classes, i.e., some distances between different classes are small while others are large. Therefore, a weighting method is used to reduce this problem. FS–LDA is an iterative procedure for subspace selection as shown in Table 2.1.

Table 2.1. Fractional–Step Linear Discriminant Analysis

Input: Training samples $\vec{x}_{i;j}$ in R^L, where *i* denotes the *i*th class ($1 \le i \le c$) and *j* denotes the *j*th sample in the *i*th class ($1 \le j \le n_i$), and the dimension L' ($L' < L$) of the projected subapce.

Output: Linear projection matrix U^* in $R^{L \times L'}$.

```
1.  Set $U = I_{L \times L}$ (the identity matrix)
2.  Calculate the mean of the ith class $\vec{m}_i = (1/n_i)\sum_{j=1}^{n_i} \vec{x}_{i;j}$.
3.  for $k = L$ to $(L'+1)$ with step $-1$ {
4.      for $l = 0$ to $(r-1)$ with step 1 {
5.          Calculate $\vec{y}_{i;j} = A^l U^T \vec{x}_{i;j}$, where $A$ is a diagonal matrix. The kth
            entry of $A$ is less than 1 and the other entries are 1;
6.          Calculate $\vec{\mu}_i = (1/n_i)\sum_{j=1}^{n_i} \vec{y}_{i;j}$ in the projected subspace;
7.          Calculate $S_b = (1/n)\sum_{i=1}^{c}\sum_{j=1}^{c} w\left(d^{(ij)}\right)\left(\vec{\mu}_i - \vec{\mu}_j\right)\left(\vec{\mu}_i - \vec{\mu}_j\right)^T$, where
            $w(\cdot)$ is the weighting function and $d^{(pij)} = \left\| \vec{m}_i - \vec{m}_j \right\|$. Usually,
            $w(\cdot)$ is set as a polynomial like function with degree less than $-3$
            (in our experiments, we set it as $-8$).
8.          Compute the first $k$ eigenvectors $\Psi = [\vec{\varphi}_1, \vec{\varphi}_2, \cdots \vec{\varphi}_k]$ of $S_b$
            associated with the largest k eigenvalues. Set $U \leftarrow U\Psi$.
9.      }//for in line 4.
10. Discard the last column of $U$.
11. }//for in line 3.
```

- Loog et al. [98][96] proposed another way to deal with the *class separation* problem by combining a weighting function with LDA and then resulted in aPAC,

$$J_{aPAC} = \sum_{i=1}^{c} \sum_{j=1}^{c} q_i q_j \omega(d_{ij}) S_w^{-1/2} (\vec{m}_i - \vec{m}_j)(\vec{m}_i - \vec{m}_j)^T S_w^{-1/2}, \tag{2.17}$$

where $d_{ij} = \sqrt{(\vec{m}_i - \vec{m}_j)^T S_w^{-1} (\vec{m}_i - \vec{m}_j)}$; $q_i = n_i / \sum_{k=1}^{c} n_k$ is the prior probability of the ith class; S_w is the within class scatter matrix defined in (2.10); $\vec{m}_i$ is the mean vector of samples in the ith class; n_i is the number of samples in the ith class ($1 \le i \le c$); c is the number of classes of the training set; $\omega(x) = (1/2x^2) \operatorname{erf}(x/2\sqrt{2})$ is the weighting function. The projection matrix is $U^* = S_w^{-1/2} \Phi$, where Φ is the matrix of leading eigenvectors of J_{aPAC} with the largest eigenvalues.

This combination J_{aPAC} approximates the mean classification accuracy (one minus Bayes error). The benefit of this method is the projection matrix can be obtained by the eigenvalue decomposition.

The weighting function based methods are not effective for the class separation problem, because it is not clear how to select an optimal weighting function. Although Loog et al. [98][96] considered the Bayesian error in aPAC, it fails to deal with the heteroscedastic problem. Even for the homoscedastic case, aPAC is also not optimal, because it only approximats to the Bayesian error. The synthetic data based test, shown in Figure 2.7 and Figure 2.8, demonstrates that FS–LDA and aPAC do not often find the optimal projection direction for classification.

Small Sample Size Problem

In many practical applications, especially in biometric research, discriminant models encounter the SSS problem [38][49][139][123][19][175][55][173][174][180], because the number of training samples is less than the dimension of the feature space. To deal with this problem, a number of algorithms were proposed. For example,

- Friedman [38] proposed the regularized discriminant analysis (RDA), which is a classification tool to smooth out the effects of ill– or poorly–conditioned covariance estimates due to the lack of training samples. RDA is a combination of ridge–shrinkage [20], LDA, and quadratic discriminant analysis (QDA) [30][39]. It provides a great number of regularization alternatives. In RDA, the regularized class covariance matrix is defined as

$$\bar{S}_i(\lambda,\gamma)=(1-\gamma)\frac{(1-\lambda)S_i+\lambda S}{(1-\lambda)n_i+\lambda n}+\frac{\gamma}{p}\operatorname{tr}\left(\frac{(1-\lambda)S_i+\lambda S}{(1-\lambda)n_i+\lambda n}\right)I, \tag{2.18}$$

where S_i is the covariance matrix of the i^{th} class; S is the covariance matrix of all samples; I is the identity matrix in $R^{p\times p}$; n_i is the number of samples of the i^{th} class; n is the number of all samples; p is the dimension of the original high dimensional feature space; and $0\le\lambda,\gamma\le 1$ are regularization parameters, which are chosen to jointly minimize an unbiased estimate of future misclassification risk through the cross validation [72] or Boostrapping [33]. In RDA, the class discriminant score for classification is

$$d_i(\vec{x})=(\vec{x}-\vec{m}_i)^T\,\bar{S}_i^{-1}(\lambda,\gamma)(\vec{x}-\vec{m}_i)+\log\left|\bar{S}_i(\lambda,\gamma)\right|-2\log\frac{n_i}{n}, \tag{2.19}$$

where $|\cdot|\square\det(\cdot)$ and $\vec{m}_i$ is the mean vector of samples in the i^{th} class. The class label of $\vec{x}$ is $\arg\min_i d_i(\vec{x})$.

Furthermore, RDA can also be used for subspace selection by introducing the reduced rank step for the sum of $d_i(\vec{x})$ over all training samples as

$$\begin{aligned}J_R=\sum_{i=1}^{c}\sum_{j=1}^{n_i}&(\vec{x}_{i;j}-\vec{m}_i)^T U\left(U^T\bar{S}_i(\lambda,\gamma)U\right)^{-1}U^T(\vec{x}_{i;j}-\vec{m}_i)\\&+\log\left|U^T\bar{S}_i(\lambda,\gamma)U\right|,\end{aligned} \tag{2.20}$$

where $|\cdot|\square\det(\cdot)$. The projection matrix is obtained by minimizing J_R over U. When $\bar{S}_i(\lambda,\gamma)=\bar{S}_j(\lambda,\gamma)$ for all $i\ne j$, RDA for subspace selection reduces to the regularized LDA (R–LDA).

- Hastie et al. [49] viewed LDA as multivariate linear regression [106] and used the penalized least squares regression [106] to reduce the SSS problem.
- Swets and Weng [139] introduced PCA as the pre–processing step in LDA for face recognition (PCA+LDA). The SSS problem is avoided if the PCA subspace has a small enough dimension. PCA+LDA is one of the most popular methods to deal with the SSS problem in biometric research. It has been being effective in many empirical demonstrations [139][4][89]. It is

also easy to be implemented as there are only two main steps: 1) conduct PCA on training samples and 2) conduct LDA on the PCA pre–processed data. PCA+LDA achieves top level performance in many applications, such as face recognition [4][88][89], gait recognition [45][46][92], and image retrieval [139]. The drawbacks [19][175] of this method are that the classification performance is sensitive to the number of features selected by PCA and PCA discards some discriminative information.

- Raudys and Duin [123] applied the pseudo–inverse [44] to the covariance matrix in LDA as P–LDA and this utilization can reduce the SSS problem in LDA. The projection matrix is obtained by maximizing

$$J_P = \operatorname{tr}\left(\left(U^T S_w U\right)^+ \left(U^T S_b U\right)\right), \tag{2.21}$$

  where A^+ means the pseudo–inversion of A, as defined in [44].

- Chen et al. [19] claimed that the most discriminative dimensions are preserved in the null space V (the complement of the range space, i.e., $S_w V = 0$) of the within class scatter matrix S_w when the dimension of the feature space is much higher than the number of training samples. First, they project all training samples to the null space of S_w. Second, PCA is utilized on the projected samples to select the projection direction for classification. To reduce the time complexity of this method, Cevikalp et al. [18] developed the discriminative common vectors scheme.
- Yu and Yang [175] claimed that the null space V (or the complement of the range space, i.e., $S_b V = 0$) of the between class scatter matrix S_b contains little discriminative information. Therefore, they first remove the null space of S_b and then select the projection matrix to minimize the within class scatter matrix S_w.
- Howland and Park [55] introduced the generalized singular value decomposition (GSVD) [44] to reduce the SSS problem in LDA. The detailed procedure is listed in Table 2.2.

Table 2.2.
Linear Discriminant Analysis via Generalized Singular Value Decomposition

Input: Training samples $\vec{x}_{i;j}$ in R^L, where *i* denotes the *i*th class ($1 \le i \le c$) and

	j denotes the j^{th} sample in the i^{th} class ($1 \le j \le n_i$), and the dimension L' ($L' < L$) of the projected subapce.
	Output: Linear projection matrix U^* in $R^{L \times L'}$.
1.	Calculate H_b and H_w according to $H_b = \left[\sqrt{n_i}\left(\vec{m}_i - \vec{m}\right)\right] \in R^{L \times c}$ and $H_b = \left[\left(x_{i;j} - \vec{m}_i\right)\right] \in R^{L \times n}$, where i and j vary over all classes and all samples in each class, separately.
2.	Computer the complete orthogonal decomposition $P^T K Q = \begin{bmatrix} R & 0 \\ 0 & 0 \end{bmatrix}$, where $K = \begin{bmatrix} H_b^T \\ H_w^T \end{bmatrix} \in R^{(c+n) \times L}$ and the rank of K is t.
3.	Conduct the singular value decomposition on $P(1:c,1:t)$, i.e., $U^T P(1:c,1:t) V = \Lambda$.
4.	Let $A = Q\begin{pmatrix} R^{-1}V & 0 \\ 0 & I \end{pmatrix}$. The projection matrix U^* is the first L' $(L' \le c-1)$ columns of A.

- Ye and Li [174] combined the orthogonal triangular decomposition [44], or briefly the QR decomposition, with LDA to reduce the SSS problem. Compared with LDA/GSVD, LDA/QR has lower time and space complexities. The only difference between PCA+LDA and LDA/QR is the first stage: PCA+LDA applies PCA to the covariance matrix of total training samples, while LDA/QR conducts the QR decomposition on a small matrix involving the class means. The detailed description of LDA/QR is given in Table 2.3.
- Recently, Zhang et al. [180] reformulated LDA based on the statistical learning theory by defining the following regularized function to reduce the SSS problem:

$$\vec{u}^* = \arg\max_{\vec{u}} \left(\frac{1}{n} \sum_{i=1}^{c} \sum_{j=1}^{n_i} \left(y_{i;j} - \vec{u}^T \vec{x}_{i;j} \right)^2 + \lambda \vec{u}^T \vec{u} \right), \tag{2.22}$$

  where $c = 2$; $y_{i;j}$ is the label identifying the class of the sample $\vec{x}_{i;j}$; λ is a small value; and $\vec{u}^T\vec{u}$ is the regularization term. Zhang et al. then extended (2.22) to multiclass problems and proved the equivalence between (2.22) and the regularized version of LDA and reported (2.22) can reduce the SSS problem. This extension has also been recognized by Gallinari et al. [42] and Hastie et al. [49].

Table 2.3. Linear Discriminant Analysis via QR Decomposition

Input:	Training samples $\vec{x}_{i;j}$ in R^L, where i denotes the ith class ($1 \le i \le c$) and j denotes the jth sample in the ith class ($1 \le j \le n_i$), and the dimension L' ($L' < L$) of the projected subapce.
Output:	Linear projection matrix $U*$ in $R^{L \times L'}$.
1.	Calculate H_b and H_w according to $H_b = \left[\sqrt{n_i}\left(\vec{m}_i - \vec{m}\right)\right] \in R^{L \times c}$ and $H_b = \left[\left(x_{i;j} - \vec{m}_i\right)\right] \in R^{L \times n}$, where i and j vary over all classes and all samples in each class, separately.
2.	Apply QR decomposition on H_b as $H_b = QRE$, where Q, R, and E are in $R^{L \times t}$, $R^{t \times c}$, and $R^{c \times c}$, respectively. Here t is the rank of H_b.
3.	Let $\overline{S}_b = RR^T$ and $\overline{S}_w = Q^T H_w H_w^T Q$. Let $\Phi = \left[\phi_i\right]_{1 \le i \le t}$ be the first t eigenvectors of $\overline{S}_b^{-1}\overline{S}_w$ associated with the first t smallest eigenvalues.
4.	The projection matrix $U*$ is defined by $U* = Q\Phi$.

Several LDA Alternatives

Several alternatives to LDA are obtained by varying the objective function defined in (2.11) for LDA. That is the projection matrix $U*$ can be obtained by maximizing the following criteria,

$$J_1 = \operatorname{tr}\left(\left(U^T S_2 U\right)^{-1}\left(U^T S_1 U\right)\right) \tag{2.23}$$

$$J_2 = \log\left|U^T S_1 U\right| - \log\left|U^T S_2 U\right| \tag{2.24}$$

$$J_3 = \operatorname{tr}\left(U^T S_1 U\right) - \lambda \operatorname{tr}\left(U^T S_2 U\right) \tag{2.25}$$

$$J_4 = \operatorname{tr}\left(U^T S_1 U\right) \big/ \operatorname{tr}\left(U^T S_2 U\right) \tag{2.26}$$

where $|\cdot| \triangleq \det(\cdot)$; the pair $\{S_1, S_2\}$ could be $\{S_b, S_w\}$, $\{S_b, S_t\}$, or $\{S_t, S_w\}$; and λ in J_3 is a Lagrange multiplier. The criterion J_1 is the traditional definition of LDA as shown in (2.11). The maximization of J_1 is equivalent to the maximization of $\operatorname{tr}\left(U^T S_1 U\right)$ with the constraint $U^T S_2 U = I$, where I is the identity matrix. In J_2, when $U^T S_b U$ is not full rank, we cannot set S_1

equal to S_b, due to $\det(S_b)=0$. The maximization of J_2 is equal to the maximization of J_1 as shown in [39].

Nonparametric Model based Discriminant Analysis

In this Section, we review some important nonparametric based discriminant analysis algorithms.

- Hastie et al. [52] viewed LDA as a multivariate linear regression and generalized LDA by replacing the multivariate linear regression with a multivariate nonparametric regression. They named the nonparametric generalization of LDA as flexible discriminant analysis. Furthermore, they also [50] introduced GMM to the flexible discriminant analysis to model complex distributions.
- Fukunaga and Mantock [40] extended the between class scatter matrix S_b and the within class scatter matrix S_w from the parametric version to the nonparametric version. With this extension, LDA is generalized to the nonparametric discriminant analysis (NDA). Using this generalization, more features can be selected for classification when $c-1$ features are not enough. Meanwhile, the assumption that each class has a Gaussian distribution is dropped.
- Buturovic [16] used the *k*–nearest–neighbour based Bayes error minimization criterion to select the discriminative subspace. Lotlikar and Kothari [94] minimized the Bayes error in the projected subspace when each class is modelled by a hyper sphere and each class has its own mean. By modelling each class density function via a kernel estimator, this method can be utilized for practical applications. Liu et al. [90] used a stochastic gradient algorithm to obtain linear representations for classification based on a new defined optimization criterion which utilizes the between class distance and within class distance information through a nonparametric way.

Kernel Discriminant Analysis

The nonlinear extension of LDA, or the kernel LDA (KDA) [109][104][129], is to solve the generalized eigenvalue decomposition

$$U_\phi{*} = \arg\max_{U_\phi} \operatorname{tr}\left(\left(U_\phi^T S_t^\phi U_\phi\right)^{-1}\left(U_\phi^T S_b^\phi U_\phi\right)\right) \tag{2.27}$$

in the higher dimensional space, where the between class scatter matrix S_b and the total class scatter matrix S_t in the higher dimensional space are defined as

$$\begin{cases} S_b^\phi = \dfrac{1}{n}\displaystyle\sum_{i=1}^{c} n_i\left(\vec{m}_{\phi,i}-\vec{m}_\phi\right)\left(\vec{m}_{\phi,i}-\vec{m}_\phi\right)^T \\ S_t^\phi = \dfrac{1}{n}\displaystyle\sum_{i=1}^{c}\sum_{j=1}^{n_i}\left(\phi\left(\vec{x}_{i;j}\right)-\vec{m}_\phi\right)\left(\phi\left(\vec{x}_{i;j}\right)-\vec{m}_\phi\right)^T \end{cases} \tag{2.28}$$

where $\vec{m}_\phi = (1/n)\sum_{i=1}^{c}\sum_{j=1}^{n_i}\phi\left(\vec{x}_{i;j}\right)$, $\vec{m}_{\phi;i} = (1/n_i)\sum_{j=1}^{n_i}\phi\left(\vec{x}_{i;j}\right)$, and $n = \sum_{i=1}^{c} n_i$.

Similar to KPCA, each column in U_ϕ is also a linear combination of all training samples in the higher dimensional space, i.e., $U_\phi = X_\phi \Lambda$, where Λ is a matrix to store linear combination coefficients as it is defined in KPCA and X_ϕ is defined as $X_\phi = \left[\phi\left(\vec{x}_{1;1}\right), \phi\left(\vec{x}_{1;2}\right), \cdots, \ \phi\left(\vec{x}_{2;1}\right), \cdots, \phi\left(\vec{x}_{c;n_c}\right)\right]$.

By replacing U_ϕ in $U_\phi{*} = \arg\max_{U_\phi} \operatorname{tr}\left(\left(U_\phi^T S_t^\phi U_\phi\right)^{-1}\left(U_\phi^T S_b^\phi U_\phi\right)\right)$ with $X_\phi \Lambda$, we have

$$\Lambda{*} = \arg\max_{\Lambda} \operatorname{tr}\left(\left(\Lambda^T K_t \Lambda\right)^{-1}\left(\Lambda^T K_b \Lambda\right)\right) \tag{2.29}$$

where K_b and K_t are defined as

$$K_b = K\left(W-M\right)\left(W-M\right)^T K \tag{2.30}$$

$$K_t = K\left(I-M\right)\left(I-M\right)^T K\,. \tag{2.31}$$

where $K = \left[k\left(\vec{x}_i, \vec{x}_j\right)\right] = \left[\left\langle \phi\left(\vec{x}_i\right), \phi\left(\vec{x}_j\right)\right\rangle\right] \in R^{n\times n}$ is the kernel Gram matrix; $k(\cdot,\cdot)$ is a kernel function [128][129]; $W = \left[W_l\right]_{1\le l\le c} \in R^{n\times n}$ is a block diagonal matrix; all entries in $W_l \in R^{n_i \times n_i}$ are $1/n_i$; and all entries in $M = \left[m_{ij}\right]_{1\le i;\, j\le n} \in R^{n\times n}$ are $1/n$.

After obtaining the linear combination coefficients, we can project a given sample $\vec{z}$ to the subspace constructed by KDA through

$$\begin{aligned} \left(X_{\phi}\Lambda\right)^{T}\phi\left(\vec{z}\right) &= \Lambda^{T}X_{\phi}^{T}\phi\left(\vec{z}\right) \\ &= \Lambda^{T}\left[k\left(\vec{x}_{1;1},\vec{z}\right),k\left(\vec{x}_{1;2},\vec{z}\right),\cdots,k\left(\vec{x}_{c;n_c},\vec{z}\right)\right]. \end{aligned} \tag{2.32}$$

where $X_{\phi}=[\phi\left(\vec{x}_i\right)]_{1\le i\le n}$.

General Averaged Divergences Analysis

If different classes are assumed to be sampled from Gaussian densities with different expected values but identical covariances, then LDA maximizes the mean value of the KL divergences between different pairs of densities. We propose a framework, the *General Averaged Divergences Analysis*, for choosing a discriminative subspace by: 1) generalizing the distortion measure from the KL divergence to the Bregman divergence[2], and 2) generalizing the arithmetic mean to the generalized mean function [48]. Based on this framework, we can develop a method to reduce the heteroscedastic problem, the unimodel problem, and the class separation problem, simultaneously.

Bregman Divergence

Definition 2.1 (Bregman Divergence): Let $U_{map}: S \to R$ be a C^1 convex function defined on a closed convex set $S \subseteq R^+$. The first derivative of U_{map} is U'_{map}, which is a monotonic function. The inverse function of U'_{map} is $\xi = \left(U'_{map}\right)^{-1}$. The probability density for the samples in ith class is $p_i(\vec{x}) = p(\vec{x} \mid y = i)$, where $y = i$ means the sample $\vec{x}$ is sampled from the ith class. The difference, as shown in Figure 2.3, at $\xi(p_j(\vec{x}))$ between the function U_{map} and the tangent line to U_{map} at $\left(\xi(p_i(\vec{x})), U_{map}\left(\xi(p_i(\vec{x}))\right)\right)$ is given by:

$$\begin{aligned} d\left(\xi(p_i(\vec{x})), \xi(p_j(\vec{x}))\right) = & \left\{U_{map}\left(\xi(p_j(\vec{x}))\right) - U_{map}\left(\xi(p_i(\vec{x}))\right)\right\} \\ & - p_i(\vec{x})\left\{\xi(p_j(\vec{x})) - \xi(p_i(\vec{x}))\right\}. \end{aligned} \tag{2.33}$$

Based on (2.33), the *Bregman divergence* for $p_i(\vec{x})$ and $p_j(\vec{x})$ is

$$D\left(p_i(\vec{x}) \,\|\, p_j(\vec{x})\right) = \int d\left(\xi(p_i(\vec{x})), \xi(p_j(\vec{x}))\right) d\mu, \tag{2.34}$$

[2] There are many distortion measures, such as Bregman divergence, Amari's α–divergence, and Csiszar's φ–divergence. The Bregman divergence is selected as the distortion measure in this thesis.

where $d\mu$ (i.e., $d\mu(\vec{x})$) is the Lebesgue measure. The right–hand side of (2.34) is also called the *U–divergence* [110]. Because U_{map} is a convex function, $d\left(\xi\left(p(\vec{x})\right),\xi\left(q(\vec{x})\right)\right)$ is non–negative. Consequently, Bregman divergence is non–negative. Because $d\left(\xi\left(p(\vec{x})\right),\xi\left(q(\vec{x})\right)\right)$ is in general not symmetric, Bregman divergence is also not symmetric. Detailed information about the Bregman divergence can be found in [110].

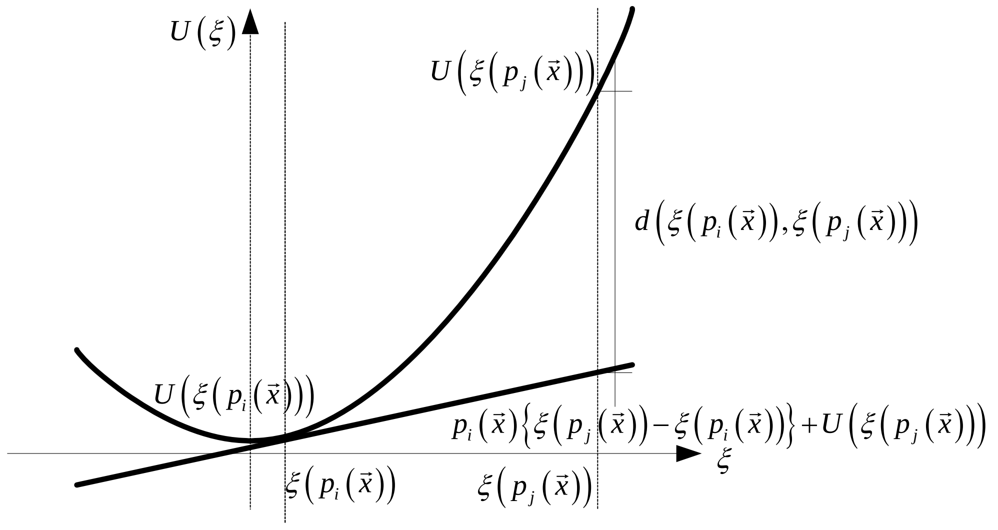

$U(\xi)$
$U\left(\xi\left(p_j(\vec{x})\right)\right)$
$d\left(\xi\left(p_i(\vec{x})\right),\xi\left(p_j(\vec{x})\right)\right)$
$U\left(\xi\left(p_i(\vec{x})\right)\right)$
$p_i(\vec{x})\left\{\xi\left(p_j(\vec{x})\right)-\xi\left(p_i(\vec{x})\right)\right\}+U\left(\xi\left(p_j(\vec{x})\right)\right)$
ξ
$\xi\left(p_i(\vec{x})\right)$
$\xi\left(p_j(\vec{x})\right)$

Figure 2.3. The geometric setting for Bregman divergence.

If $U_{map}(x)=\exp(x)$, Bregman divergence reduces to the KL–divergence,

$$\begin{aligned} D\left(p_i(\vec{x})\,\|\,p_j(\vec{x})\right) &= \int\left(p_j(\vec{x})-p_i(\vec{x})-p_i(\vec{x})\log\frac{p_j(\vec{x})}{p_i(\vec{x})}\right)d\mu \\ &= \int p_i(\vec{x})\log\frac{p_i(\vec{x})}{p_j(\vec{x})}d\mu = KL\left(p_i(\vec{x})\,\|\,p_j(\vec{x})\right). \end{aligned} \tag{2.35}$$

Further examples can be found in [110].

For Gaussian probability density functions, $p_i(\vec{x}) \sim N(\vec{x};\vec{m}_i,\Sigma_i)$, where $\vec{m}_i$ is the mean vector of the ith class samples and Σ_i is the within class covariance matrix of the ith class, the KL divergence [21] is

$$\begin{aligned} KL\left(p_i(\vec{x}) \| p_j(\vec{x})\right) &= \int N(\vec{x};\vec{m}_i,\Sigma_i) \ln \frac{N(\vec{x};\vec{m}_i,\Sigma_i)}{N(\vec{x};\vec{m}_j,\Sigma_j)} d\vec{x} \\ &= \ln|\Sigma_j| - \ln|\Sigma_i| + \operatorname{tr}\left(\Sigma_j^{-1}\Sigma_i\right) + \operatorname{tr}\left(\Sigma_j^{-1} D_{ij}\right), \end{aligned} \tag{2.36}$$

where $D_{ij} = (\vec{m}_i - \vec{m}_j) \otimes (\vec{m}_i - \vec{m}_j)$ and $|\Sigma| \triangleq \det(\Sigma)$.

To simplify the notation we denote the KL divergence between the projected densities $p(U^T\vec{x} \mid y = i)$ and $p(U^T\vec{x} \mid y = j)$ by

$$D_U(p_i \| p_j) \triangleq D\left(p(U^T\vec{x} \mid y = i) \| p(U^T\vec{x} \mid y = j)\right). \tag{2.37}$$

General Averaged Divergences Analysis

We replace the arithmetic mean by the generalized mean [48],

$$V_\varphi(U) = \varphi^{-1}\left[\frac{\sum_{1 \le i \ne j \le c} q_i q_j \varphi\left(D_U(p_i \| p_j)\right)}{\sum_{1 \le m \ne n \le c} q_m q_n}\right], \tag{2.38}$$

where $\varphi(\cdot)$ is a strict monotonic real–valued increasing function defined on $(0, +\infty)$; $\varphi^{-1}(\cdot)$ is the inverse function of $\varphi(\cdot)$; q_i is the prior probability of the i^{th} class (usually, we set $q_i = n_i/n$ or simply set $q_i = 1/c$); $D_U(p_i \| p_j)$ is defined in (2.37); $\vec{x} \in R^L$ where R^L is the feature space containing the training samples; and $U \in R^{L \times L'}$ ($L > L'$) is the projection matrix. The general averaged divergences function measures the average of all divergences between pairs of classes in the subspace. We obtain the projection matrix U^* by maximizing the general averaged divergences function $V_\varphi(U)$ over U, for a fixed $\varphi(\cdot)$. The general optimization algorithm for subspace selection based on (2.38) is given in Table 2.4. Usually, the concavity of the averaged divergences cannot be guaranteed. To reduce the effects of local maxima [11], we choose a number of different initial projection matrices; carry out the separate optimizations; and then select the best one, which has the maximal value of $V_\varphi(U)$.

If $V_\varphi(U)$ depends only on the subspace of R^n spanned by the columns of U then U can be replaced by UC where C is a $k \times k$ matrix, chosen such that the columns of UC are orthogonal.

On setting $\varphi(x)=x$, we obtain the arithmetic mean based method for choosing a subspace,

$$U^*=\arg\max_U \sum_{1\le i\ne j\le c} \frac{q_i q_j D_U\left(p_i \,\|\, p_j\right)}{\sum_{1\le m\ne n\le c} q_m q_n} = \arg\max_U \sum_{1\le i\ne j\le c} q_i q_j D_U\left(p_i \,\|\, p_j\right). \qquad (2.39)$$

Table 2.4. General Averaged Divergences Analysis for Subspace Selection

Input: Training samples $\vec{x}_{i;j}$ in R^L, where *i* denotes the *i*th class ($1\le i\le c$) and *j* denotes the *j*th sample in the *i*th class ($1\le j\le n_i$), the dimension L' ($L'<L$) of the projected subspace, and *M* is the maximum number of different initial values for the projection matrix.

Output: Optimal linear projection matrix U^* in $R^{L\times L'}$.

```
1.  for $m=1:M$ {
2.      Randomly initialize $U_t^m$ ($t=1$), i.e., all entries of $U_1^m$ are random numbers.
3.      while $\left|V_\varphi\left(U_t^m\right)-V_\varphi\left(U_{t-1}^m\right)\right|>\varepsilon$ ($\varepsilon=10^{-6}$), do{
4.          Conduct the gradient steepest ascent algorithm[3] to maximize the averaged divergences defined in (2.38): $U_t^m \leftarrow U_{t-1}^m+\kappa\cdot\partial_U V_\varphi\left(U_{t-1}^m\right)$. Here, $\kappa$, a small value (e.g., 0.0001), is the learning rate.
5.          $t\leftarrow t+1$
6.      }//while in line 3
7.  }//for in line 1
8.  $U^*\leftarrow \arg\max_m V_\varphi\left(U_t^m\right)$.
```

Observation 2.1: LDA maximizes the arithmetic mean of the KL divergences between all pairs of classes, under the assumption that the Gaussian distributions $p_i(\vec{x})$ for different classes all have the same covariance matrix. The optimal

[3] The gradient steepest ascent algorithm can be replaced by other faster optimization methods, such as the conjugate gradient method, to reduce the number of iterations.

projection matrix U with respect to LDA can be obtained by maximizing a particular $V_{\varphi}(U)$, i.e.,

$$\arg\max_{U} \sum_{1 \le i \ne j \le c} q_i q_j D_U\left(p_i \| p_j\right) = \arg\max_{U} \operatorname{tr}\left(\left(U^T S_w U\right)^{-1} U^T S_b U\right). \tag{2.40}$$

Proof: See Appendix.

Example: Decell and Mayekar [29] maximized the weighted arithmetic mean of all symmetric KL divergences between all pairs of classes in the projected subspace. The weighting factor of the symmetric KL divergence between the *i*th class and the *j*th class is $q_i q_j$, where q_i is the prior probability of the *i*th class.The symmetric KL divergence is:

$$\begin{aligned} SKL\left(p_i \| p_j\right) &= \frac{1}{2} KL\left(p_i \| p_j\right) + \frac{1}{2} KL\left(p_j \| p_i\right) \\ &= \operatorname{tr}\left(\Sigma_j^{-1}\Sigma_i + \Sigma_i^{-1}\Sigma_j\right) + \operatorname{tr}\left(\left(\Sigma_j^{-1} + \Sigma_i^{-1}\right)\left(\vec{m}_i - \vec{m}_j\right)\left(\vec{m}_i - \vec{m}_j\right)^T\right). \end{aligned} \tag{2.41}$$

It follows from (2.41) that Decell and Mayekar's method also maximizes the arithmetic mean of all KL divergences.

De la Torre and Kanade [28] developed ODA based on the objective function (2.12) used in [29], but using the iterative majorization to quickly obtain a solution.

How to Deal with the Multimodal Problem [50]

Up to this point it has been assumed that the samples in a given class are sampled from a single Gaussian distribution. This assumption often fails in real–world large data sets, such as those used for multi–view face [84] and gait [176] recognition, natural image classification [156] or texture classification [133].

To overcome this limitation, each class can be modelled by a GMM. Many methods for obtaining GMMs have been described in the literature. Examples include KMeans [30], GMM with expectation–maximization (EM) [30], graph–cut [131], and spectrum clustering. Unfortunately, these methods are not adaptive, in that the number of subclusters must be specified, and some of them (e.g., EM and KMeans) are sensitive to initial values. In our algorithm we use the recently introduced GMM–EM like algorithm, proposed by Figueiredo and Jain [36], which is named the GMM–FJ method. The reasons for choosing GMM–FJ are: it finds the number of subclusters; it is less sensitive to the choice of initial values of parameters than EM; and it avoids the boundary problem[4] of the parameter space. We assume that samples in each class are sampled from a GMM and the projection matrix U can be obtained by maximizing the general averaged divergences, which measure the averaged distortion between any pair of subclusters in different classes, i.e.,

$$V_{\varphi}(U) = \varphi^{-1}\left[\frac{\sum_{1\le i\ne j\le c}\sum_{1\le k\le C_i}\sum_{1\le l\le C_j} q_i^k q_j^l \varphi\left(D_U\left(p_i^k \,\|\, p_j^l\right)\right)}{\sum_{1\le m\ne n\le c}\sum_{1\le s\le C_i}\sum_{1\le t\le C_j} q_m^s q_n^t}\right], \tag{2.42}$$

where q_i^k is the prior probability of the kth subcluster of the ith class; p_i^k is the sample density of the samples in the kth subcluster in the ith class; $D_U\left(p_i^k \,\|\, p_j^l\right)$ is the divergence between the kth subcluster in the ith class and the lth subcluster in the jth class.

[4] The means of each class is equal to one of the samples and the covariances are arbitrarily close to singular. For example, we have *N* samples and we use *N* Gaussians to model these samples. The mean of each Gaussian is equivalent to a sample and the covariance of each Gaussian is singular.

Geometric Mean for Subspace Selection

In LDA and ODA, the arithmetic mean of the divergences is used to find a suitable subspace into which to project the samples. The main benefit of using the arithmetic mean in LDA is that the projection matrix can be obtained by the generalized eigenvalue decomposition. However, LDA is not optimal for multiclass classification [103] because of the *class separation* problem defined in §103. Therefore, it is useful to investigate other choices of φ in (2.38) to see if better results can be obtained.

Criterion 1:
Maximization of the Geometric Mean of the Divergences

The log function is a suitable choice for φ because it increases the effects of small divergences and at the same time reduces the effects of large divergences. On setting $\varphi(x) = \log(x)$ in (2.38), the generalized geometric mean of the divergences is obtained. The required subspace U^* is given by,

$$U^* = \arg\max_{U} \prod_{1 \le i \ne j \le c} \left[D_U\left(p_i \,\|\, p_j\right) \right]^{\frac{q_i q_j}{\sum_{1 \le m \ne n \le c} q_m q_n}} . \tag{2.43}$$

It follows from the mean inequality that the generalized geometric mean is upper bounded by the arithmetic mean of the divergences, i.e.,

$$\prod_{1 \le i \ne j \le c} \left[D_U\left(p_i \,\|\, p_j\right) \right]^{\frac{q_i q_j}{\sum_{1 \le m \ne n \le c} q_m q_n}} \le \sum_{1 \le i \ne j \le c} \left(\frac{q_i q_j}{\sum_{1 \le m \ne n \le c} q_m q_n} D_U\left(p_i \,\|\, p_j\right) \right) . \tag{2.44}$$

Proof: See Appendix.

Furthermore, (2.43) emphasizes the total volume of all divergences. For example, in the special case $q_i = q_j$ for all i, j,

$$\begin{aligned} &\arg\max_{U} \prod_{1 \le i \ne j \le c} \left[D_U\left(p_i \,\|\, p_j\right) \right]^{\frac{q_i q_j}{\sum_{1 \le m \ne n \le c} q_m q_n}} = \arg\max_{U} \prod_{1 \le i \ne j \le c} \left[D_U\left(p_i \,\|\, p_j\right) \right]^{q_i q_j} \\ &= \arg\max_{U} \prod_{1 \le i \ne j \le c} D_U\left(p_i \,\|\, p_j\right) . \end{aligned} \tag{2.45}$$

Criterion 2:

Maximization of the Geometric Mean of the Normalized Divergences

We can further strengthen the effects of small divergences on subspace selection, by maximizing the geometric mean[5] of all normalized divergences[6] in the projected subspace, i.e.,

$$U^* = \arg\max_U \left[\prod_{1 \le i \ne j \le c} E_U\left(p_i \parallel p_j\right) \right]^{\frac{1}{c(c-1)}} = \arg\max_U \prod_{1 \le i \ne j \le c} E_U\left(p_i \parallel p_j\right), \tag{2.46}$$

where the normalized divergence $E_U\left(p_i \parallel p_j\right)$ between the ith class and the jth class is defined by:

$$E_U\left(p_i \parallel p_j\right) = \frac{q_i q_j D_U\left(p_i \parallel p_j\right)}{\sum_{1 \le m \ne n \le c} q_m q_n D_U\left(p_m \parallel p_n\right)}. \tag{2.47}$$

The intuition behind (2.46) is that the product of normalized divergences is large when the normalized divergences are similar to each other. Therefore, maximizing the geometric mean of the normalized divergences will tend to make them as similar as possible. The effects of the small divergences should then be emphasized.

Criterion 3:

Maximization of the Geometric Mean of all Divergences

Although criterion 2 emphasizes small divergences during optimization, direct use of the criterion is not desirable for subspace selection. This is because experiments in §233066540 show that there exists U for which all divergences become small, but all normalized divergences are comparable in size. In such case, the projection matrix U is not suitable for classification, because several classes may be severely overlapped.

To reduce this problem, we combine criterion 2 with criterion 1 into a new one. The new criterion maximizes the linear combination of: 1) the log of the

[5] In (2.46), we use the geometric mean but not the generalized geometric mean because the weights (the prior probabilities q_i) are moved to the normalized divergences as shown in (2.47). In this form, the calculations are simplified.

[6] The sum of all normalized divergences is one.

geometric mean of the divergences and 2) the log of the geometric mean of normalized divergences. This criterion is named the *Maximization of the Geometric Mean of all Divergences*, or briefly MGMD,

$$
\begin{aligned}
U^* &= \arg\max_U \left\{ \begin{array}{l} \alpha \log \left[\prod_{1 \le i \ne j \le c} E_U\left(p_i \,\|\, p_j\right) \right]^{\frac{1}{c(c-1)}} \\ +(1-\alpha)\log \prod_{1 \le i \ne j \le c} \left[D_U\left(p_i \,\|\, p_j\right) \right]^{\frac{q_i q_j}{\sum_{1 \le m \ne n \le c} q_m q_n}} \end{array} \right\} \\
&= \arg\max_U \left\{ \sum_{1 \le i \ne j \le c} \log D_U\left(p_i \,\|\, p_j\right) - \eta \log\left(\sum_{1 \le i \ne j \le c} q_i q_j D_U\left(p_i \,\|\, p_j\right) \right) \right\},
\end{aligned}
\tag{2.48}
$$

where $0 < \alpha < 1$ is the linear combination coefficient to integrate the criterion 1 with 2; and $\eta = \dfrac{\alpha c(c-1) \sum_{1 \le m \ne n \le c} q_m q_n}{c(c-1)(1-\alpha) + \alpha \sum_{1 \le m \ne n \le c} q_m q_n}$. The supremum of η is $c(c-1)$ and the infimum of η is 0. When $\alpha = 0$ (or $\eta = 0$), (2.48) reduces to (2.43); and when $\alpha = 1$ (or $\eta = c(c-1)$), (2.48) reduces to (2.46). The rational of the combination coefficient α trades off the geometric mean of divergences of different pairs of classes against the geometric mean of normalized divergences of different pairs of classes. The first part emphasizes the total volume of all divergences; increases effects of small divergences; and reduces effects of large divergences. The second part further strengthens the effects of small divergences and weakens theeffects of large divergences. Therefore, by tuning the parameter α, we can balance the impacts of selected subspace on the total volume of all divergences, the impacts of large divergences, and the impacts of small ones.

Deduction 2.1: See Appendix.

Based on (2.48) and (2.42), we directly extend MGMD to the multimodal case, as the *Multimodal extension of the Maximization of the Geometric Mean of all Divergences* (M–MGMD),

$$
U^* = \arg\max_U \left\{ \begin{array}{l} \sum_{1 \le i \ne j \le c} \sum_{1 \le k \le C_i} \sum_{1 \le l \le C_j} \log D_U\left(p_i^k \,\|\, p_j^l\right) \\ -\eta \log\left(\sum_{1 \le i \ne j \le c} \sum_{1 \le k \le C_i} \sum_{1 \le l \le C_j} q_i^k q_j^l D_U\left(p_i^k \,\|\, p_j^l\right) \right) \end{array} \right\}. \tag{2.49}
$$

In the next Section, we discuss subspace selection based on the choice of the KL divergence in (2.48) and (2.49).

Kullback–Leibler Divergence based Subspace Selection

In the last Section, we developed a framework MGMD/M–MGMD for subspace selection based on the geometric mean of divergences and normalized divergences. In this Section, we combine the KL divergence with MGMD/M–MGMD as an example for practical applications. The multimodal problem [50] is carefully studied and finally we kernelize the proposed subspace selection method. Experimental studies are given in §50, §50, and §50.

MGMD and KL Divergence for Subspace Selection

By combining the KL divergence defined in (2.36) and MGMD defined in (2.48), we obtain the *Maximization of the Geometric Mean of all KL Divergences* (MGMKLD),

$$U^* = \arg\max_U L(U), \tag{2.50}$$

where $L(U)$ is defined by

$$L(U) = \sum_{1 \le i \ne j \le c} \log KL_U\left(p_i \,||\, p_j\right) - \eta \log\left(\sum_{1 \le i \ne j \le c} q_i q_j KL_U\left(p_i \,||\, p_j\right)\right), \tag{2.51}$$

and $KL_U\left(p_i \,||\, p_j\right)$ is the KL divergence between the ith class and the jth class in the projected subspace,

$$\begin{aligned} KL_U\left(p_i \,||\, p_j\right) = \frac{1}{2}\log\left|U^T\Sigma_j U\right| - \log\left|U^T\Sigma_i U\right| \\ + \operatorname{tr}\left(\left(U^T\Sigma_j U\right)^{-1}\left(U^T\Sigma_i U\right)\right) + \operatorname{tr}\left(\left(U^T\Sigma_j U\right)^{-1} U^T D_{ij} U\right) \end{aligned} \tag{2.52}$$

To better understand MGMKLD, we need to define the Stiefel manifold and the Grassmann manifold.

Definition 2.2: The Stiefel manifold [10] $St(n,r)$ for $n \ge r$ is defined as a set of all $n \times r$ matrices with orthonormal columns, i.e., $St(n,r) = \{U \in R^{n \times r} : U^T U = I_r\}$. $St(n,r)$ is a sub–manifold of $R^{n \times r}$ of real dimension $2n \times r - r^2$. Two elements U_1 and U_2 in $St(n,r)$ are said to be equivalent if their columns span the same subspace, i.e., $U_1 = U_2 Q$ for some orthogonal matrix $Q \in R^{r \times r}$.

Table 2.5. Optimization procedure for MGMKLD

Input: Training samples $\vec{x}_{i;j}$ in R^n, where i denotes the ith class ($1 \le i \le c$) and j denotes the jth sample in the ith class ($1 \le j \le n_i$), the dimension k ($k < n$) of the selected subspace, the maximum number M of different initial values for the projection matrix U, the learning rate κ (a small value), the combination factor η, and a small value ε as the convergence condition.

Output: An estimate U^* in $R^{n \times k}$ of the optimal projection matrix.

```
1.  for $m = 1 : M$ {
2.      Initialize $U_t^m$ ($t = 1$) randomly.
3.      while $\left| L\left(U_t^m\right) - L\left(U_{t-1}^m\right) \right| > \varepsilon$ ($\varepsilon = 10^{-6}$),
        where $L\left(U_t^m\right)$ and $KL_{U_t^m}\left(p_i \| p_j\right)$ are defined in (2.51) and (2.52),
        respectively.
        do{
4.          Conduct the gradient steepest ascent step:
            $U_t^m \leftarrow U_{t-1}^m + \kappa \cdot \partial_U L\left(U_{t-1}^m\right)$, where $\partial_U L\left(U_t^m\right)$ is defined in (2.53).
5.          $t \leftarrow t + 1$.
5.      }//while in line 3
6.  }//for in line 1
7.  $U \leftarrow \arg\max_m V_\varphi\left(U^m\right)$
8.  Ortho–normalization Step: $U^* \leftarrow \text{ortho-normalize}\left(U\right)$.
```

Definition 2.3: The Grassmann manifold [10] is the quotient space of $St\left(n,r\right)$ with respect to the above equivalent relation. Each element in the Grassmann manifold $Gr\left(n,r\right)$ is an equivalent class in $St\left(n,r\right)$. $Gr\left(n,r\right)$ is a set of all r–dimensional vector subspaces of R^n [7].

As mentioned in the Table 2.4, the procedure for the maximization of the general averaged divergences, we prove the projection matrix U is invariant to the

[7] http://mathworld.wolfram.com/GrassmannManifold.html

ortho–normalization operator for $L(U)$, i.e., U stays in the Grassmann manifold. In other words, $L(U)$ will depend only on the subspace defined by U.

Claim 2.1: The ortho–normalization operation does not change the value of the objective function $L(U)$ of MGMKLD defined in (2.51).

Proof: See Appendix.

To obtain the optimization procedure for MGMKLD based on Table 2.4, we need the first order derivative of $L(U)$,

$$\begin{aligned}
&\partial_U L(U) \\
&= \sum_{1\le i\ne j\le c} KL_U^{-1}\left(p_i \,\|\, p_j\right)\partial_U KL_U\left(p_i \,\|\, p_j\right) \\
&\quad -\eta\left(\sum_{1\le m\ne n\le c} q_m q_n KL_U\left(p_m \,\|\, p_n\right)\right)^{-1}\left(\sum_{1\le i\ne j\le c} q_i q_j \partial_U KL_U\left(p_i \,\|\, p_j\right)\right),
\end{aligned} \tag{2.53}$$

where the first order derivative of $KL_U\left(p_i \,\|\, p_j\right)$ is given by

$$\begin{aligned}
&\partial_U KL_U\left(p_i \,\|\, p_j\right) \\
&= \mathbf{\Sigma}_j U\left(U^T\Sigma_j U\right)^{-1} - \Sigma_i U\left(U^T\Sigma_i U\right)^{-1} + \left(\Sigma_i + D_{ij}\right)U\left(U^T\Sigma_j U\right)^{-1} \\
&\quad - \Sigma_j U\left(U^T\Sigma_j U\right)^{-1} U^T\left(\Sigma_i + D_{ij}\right)U\left(U^T\Sigma_j U\right)^{-1}.
\end{aligned} \tag{2.54}$$

Multimodal Extension of MGMKLD

In many situations, the distribution of each class is not Guassian. It is reasonable to use GMMs to fit each class. The advantages of introducing GMM to discriminative subspace selection are described in §0. In this Section, we combine GMM with MGMKLD as the multimodal extension of MGMKLD (M–MGMKLD). The combination, M–MGMKLD, is obtained from (2.36) and (2.49), as

$$\begin{aligned}
U^* &= \arg\max_U L_{GMM}(U) \\
&= \arg\max_U \sum_{1\le i\ne j\le c}\sum_{1\le k\le C_i}\sum_{1\le l\le C_j} \log\left[KLU_{\mathrm{w}}\left(p_i^k \,\|\, p_j^l\right)\right] \\
&\qquad -\eta\log\left[\sum_{1\le i\ne j\le c}\sum_{1\le k\le C_i}\sum_{1\le l\le C_j} q_i^k q_j^l\left[KL_U\left(p_i^k \,\|\, p_j^l\right)\right]\right].
\end{aligned} \tag{2.55}$$

Claim 2.2: The ortho–normalization operation does not change the value of the objective function $L_{GMM}(U)$ of M–MGMKLD.

Proof. This claim is proved in the same way as the Claim 2.1. ■

Table 2.6. Optimization procedure for M–MGMKLD.

Input: Training samples $\vec{x}_{i;j}$ in R^n, where i denotes the ith class ($1 \le i \le c$) and j denotes the jth sample in the ith class ($1 \le j \le n_i$), the dimension k ($k < n$) of the selected subspace, the maximum number M of different initial values for U, the learning rate κ (a small value), the combination factor η, and a small value ε as the convergence condition.

Output: Linear projection matrix U^* in $R^{n \times k}$.

```
1.  Conduct the GMM–FJ to cluster samples in each class and obtain the
    corresponding covariance matrix $\Sigma_i^k$ and mean value $\vec{m}_i^k$, where
    $1 \le i \le c$ and $1 \le k \le C_i$, $C_i$ is the number of clusters of the $i$th class.
2.  for $m = 1:M$ {
3.      Initialize $U_1^m$ randomly.
4.      while $\left|L_{GMM}\left(U_t^m\right) - L_{GMM}\left(U_{t-1}^m\right)\right| > \varepsilon$ ($\varepsilon = 10^{-6}$),
        where $L_{GMM}(U)$ is defined in (2.55).
        do{
5.          Conduct the gradient steepest step: $U_t^m \leftarrow U_{t-1}^m + \kappa \cdot \partial_U L_{GMM}\left(U_{t-1}^m\right)$,
            where $\partial_U L_{GMM}(U)$ is defined in (2.56).
6.      }//while in line 4
7.      $U \leftarrow \arg\max_m V_\varphi\left(U^m\right)$.
8.  }//for in line 2
9.  Ortho–normalization Step: $U^* \leftarrow \text{ortho-normalize}(U)$.
```

With Claim 2.2 and Table 2.4, we can obtain the optimization procedure for M–MGMKLD. We only need to mention the first order derivative of the objective function $L_{GMM}(U)$ of M–MGMKLD,

$$\begin{aligned}\partial_U L_{GMM}(U) = &\sum_{1\le i\ne j\le c}\sum_{1\le k\le C_i}\sum_{1\le l\le C_j} KL_U^{-1}\left(p_i^k \,\|\, p_j^l\right)\partial_U KL_U\left(p_i^k \,\|\, p_j^l\right)\\ &-\eta\left(\sum_{1\le m\ne n\le c}\sum_{1\le s\le C_i}\sum_{1\le t\le C_j} q_m^s q_n^t KL_U\left(p_m^s \,\|\, p_n^t\right)\right)^{-1}\\ &\left(\sum_{1\le i\ne j\le c}\sum_{1\le k\le C_i}\sum_{1\le l\le C_j} q_i^k q_j^l \partial_U KL_U\left(p_i^k \,\|\, p_j^l\right)\right).\end{aligned} \tag{2.56}$$

By incorporating the merits of GMM–FJ and the proposed MGMKLD model, the new method has the following benefits:

1. Experiments show that the new method reduces the *class separation* problem;
2. The new method inherits the merits of GMM–FJ. In detail, it determines the number of subclusters in each class automatically; it is less sensitive to the choice of initial values of the parameters than EM; and it avoids the boundary problem of the parameter space;
3. The new method is capable of obtaining the projection orthogonal matrix, i.e., $L_{GMM}(U)$ depends only on the subspace defined by U.

Kernel Extension of MGMKLD

In this part, we study the kernel extension of MGMKLD. Because of the success of the kernel method [129][109] in pattern classification, MGMKLD will be generalized from the low dimensional original Hilbert space to the higher dimensional Hilbert space. The generalized version is named the kernel MGMKLD (KMGMKLD).

To utilize the kernel dot product trick [129][109] for MGMKLD, we need to have the Lemma 2.1. Denote $\oplus$ as the direct sum.

Lemma 2.1: If U is a solution to MGMKLD and $U = U_x \oplus U_x^{\perp}$, then U_x is a solution to MGMKLD. We have $L(U) = L(U_x)$. Here, the column space of U_x is spanned by the samples $\left\{\vec{x}_{i;j}\right\}\big|_{1\le i\le c}^{1\le j\le n_i}$ and the column space of $U_x^{\perp}$ is the orthogonal complement of the column space of U_x.

Proof: See Appendix.

From Lemma 2.1, we know that the orthogonal complement component $U_x^{\perp}$ ($U_x = U$) does not affect the objective function $L(U)$ of MGMKLD defined in

(2.51). Consequently, we can set $U_x^{\perp} = 0$, i.e., the column space of U is spanned by the samples $\left\{\vec{x}_{i;j}\right\}\big|_{1\le i\le c}^{1\le j\le n_i}$. Based on Lemma 2.1, the kernel trick can be utilized to implement the kernel extension of MGMKLD, because $L(U)$ defined in (2.51) can be fully expressed in terms of inner products with the samples $\left\{\vec{x}_{i;j}\right\}\big|_{1\le i\le c}^{1\le j\le n_i}$ only. Without this lemma, the kernelization of MGMKLD cannot be implemented.

Herein, there is a mapping rule $\phi: R^L \mapsto R^H$ to map MGMKLD to a higher–dimensional space [109][104][129]. The samples $\vec{x}$, which are modelled by $N\left(\vec{m}_i, \Sigma_i\right)$, are mapped as

$$\vec{x}_{i;j} \to \phi\left(\vec{x}_{i;j}\right) \sim N\left(\phi\left(\vec{m}_i\right), \phi\left(\Sigma_i\right)\right) \tag{2.57}$$

$$\vec{m}_{\phi;i} = \frac{1}{n_i}\sum_{j=1}^{n_i}\phi\left(\vec{x}_{i;j}\right) \tag{2.58}$$

$$\Sigma_{\phi;i} = E\left[\left(\phi\left(\vec{x}_{i;j}\right) - \vec{m}_{\phi;i}\right)\left(\phi\left(\vec{x}_{i;j}\right) - \vec{m}_{\phi;i}\right)^T\right] \tag{2.59}$$

where $\vec{m}_{\phi;i}$ is the mean vector of the ith class in R^H and $\Sigma_{\phi;i}$ is the covariance matrix of the ith class in R^H.

Based on Lemma 2.1, we can choose the projection matrix in R^H as

$$U_\phi = \sum_{i=1}^{c}\sum_{j=1}^{n_i}\alpha_{i;j}\phi\left(\vec{x}_{i;j}\right) = \left[\phi\left(\vec{x}_{i;j}\right)\right]_{F\times n}\Lambda_{n\times F^*}, \tag{2.60}$$

where $\alpha_{i;j}$ is the linear combination coefficient to combine the training samples $\phi\left(\vec{x}_{i;j}\right)$ in R^H.

This means each column of U_ϕ is a linear combination of all $\phi\left(\vec{x}_{i;j}\right)$ by varying *i* and *j* from all classes and samples in each class.

Therefore, the KL divergence $KL_U\left(p_i\left(\phi\left(\vec{x}\right)\right) \| p_j\left(\phi\left(\vec{x}\right)\right)\right)$ in the feature space is

$$\begin{aligned} & KL_{U_\phi}\left(p_i\left(\phi\left(\vec{x}\right)\right) \| p_j\left(\phi\left(\vec{x}\right)\right)\right) \\ & = \frac{1}{2}\begin{pmatrix} \log\left|U_\phi^T\Sigma_{\phi;j}U_\phi\right| - \log\left|U_\phi^T\Sigma_{\phi;i}U_\phi\right| \\ + \operatorname{tr}\left(\left(U_\phi^T\Sigma_{\phi;j}U_\phi\right)^{-1}\left(U_\phi^T\left(\Sigma_{\phi;j} + D_{\phi;ij}\right)U_\phi\right)\right) \end{pmatrix}. \end{aligned} \tag{2.61}$$

Therefore, (2.51) becomes

$$L(U_\phi) = \sum_{1 \le i \ne j \le c} \left[\log KL_{U_\phi}\left(p_i(\phi(\vec{x})) \| p_j(\phi(\vec{x})) \right) \right] \\ - \eta \log \left[\sum_{1 \le m \ne n \le c} q_m q_n KL_{U_\phi}\left(p_m(\phi(\vec{x})) \| p_n(\phi(\vec{x})) \right) \right]. \tag{2.62}$$

The variable D_{ij} becomes

$$D_{\phi;ij} = \left(\frac{1}{n_i} \sum_{k=1}^{n_i} \phi(\vec{x}_{i;k}) - \frac{1}{n_j} \sum_{l=1}^{n_j} \phi(\vec{x}_{i;l}) \right) \left(\frac{1}{n_i} \sum_{k=1}^{n_i} \phi(\vec{x}_{i;k}) - \frac{1}{n_j} \sum_{l=1}^{n_j} \phi(\vec{x}_{i;l}) \right)^T. \tag{2.63}$$

To obtain the kernel Gram matrix [129] based representation in (2.61), we need to reformulate $U_\phi^T(U)\Sigma_{\phi;i}U_\phi$ by the kernel dot product trick[8] as:

$$U_\phi^T \Sigma_{\phi;i} U_\phi \\ = \frac{1}{n_i} \Lambda_{n \times H'}^T K_{C_i} \left(I_{C_i,C_i} - \frac{1}{n_i} 1_{C_i,C_i} \right) \left(I_{C_i,C_i} - \frac{1}{n_i} 1_{C_i,C_i} \right)^T K_{C_i}^T \Lambda_{n \times H'}, \tag{2.64}$$

Deduction 2.2: See Appendix.

where $K_{:;j}$ is the $\left(\sum_{k=1}^{i-1} n_k + j\right)^{th}$ column of the kernel Gram matrix $K = \left[\phi(\vec{x}_{i;j})\right]_{F \times n}^T \left[\phi(\vec{x}_{i;j})\right]_{F \times n} = \left[k(\vec{x}_{i;j}, \vec{x}_{i;j})\right]$, $k(\vec{x}_{i;j}, \vec{x}_{i;j})$ is the kernel function [129], $I_{C_i,C_i} \in R^{C_i,C_i}$ is the identity matrix, $1_{C_i,C_i} \in R^{C_i,C_i}$ is the unit matrix, $K_{C_i} = \left[K_{:;j}\right]_{1 \le j \le n_i}$ is composed of the columns in the kernel Gram matrix from the $\left(\sum_{k=1}^{i-1} n_k + 1\right)^{th}$ column to the $\left(\sum_{k=1}^{i} n_k\right)^{th}$ column; $\Lambda_{n \times H'}$ is the projection matrix in R^H and H' is the number of selected features in R^H.

With the kernel dot product trick we can transform $U_\phi^T D_{\phi;ij} U_\phi$ into

$$U_\phi^T D_{\phi;ij} U_\phi \\ = \Lambda_{n \times H'}^T \left(\frac{1}{n_i} K_{C_i} 1_{C_i} - \frac{1}{n_j} K_{C_j} 1_{C_j} \right) \left(\frac{1}{n_i} K_{C_i} 1_{C_i} - \frac{1}{n_j} K_{C_j} 1_{C_j} \right)^T \Lambda_{n \times H'}. \tag{2.65}$$

Deduction 2.3: See Appendix.

Therefore, we can reformulate (2.61) as:

[8] The kernel dot product trick is used to transform a linear algorithm (for classification, subspace selection, regression, and etc.) to a non–linear one by mapping the original samples in a low dimensional space into a higher dimensional space. Based on this mapping, the linear algorithm in the new space is equivalent to non–linear algorithm in the original space. The trick transforms a linear algorithm by replacing the dot product between two vectors with a kernel function.

$$KL_{ij}\left(U_\phi\right)$$

$$=\frac{1}{2}\left(\begin{array}{l}\log\left|\frac{1}{n_j}\Lambda_{n\times H'}^T\bar{K}_{C_j}\bar{K}_{C_j}^T\Lambda_{n\times H'}\right|-\log\left|\frac{1}{n_i}\Lambda_{n\times H'}^T\bar{K}_{C_i}\bar{K}_{C_i}^T\Lambda_{n\times H'}\right| \\ +\operatorname{tr}\left(\left(\frac{1}{n_j}\Lambda_{n\times H'}^T\bar{K}_{C_j}\bar{K}_{C_j}^T\Lambda_{n\times H'}\right)^{-1}\left(\Lambda_{n\times H'}^T L_{C_i,C_j}L_{C_i,C_j}^T\Lambda_{n\times H'}\right)\right) \\ +\operatorname{tr}\left(\left(\frac{1}{n_j}\Lambda_{n\times H'}^T\bar{K}_{C_j}\bar{K}_{C_j}^T\Lambda_{n\times H'}\right)^{-1}\left(\frac{1}{n_i}\Lambda_{n\times H'}^T\bar{K}_{C_i}\bar{K}_{C_i}^T\Lambda_{n\times H'}\right)\right)\end{array}\right), \tag{2.66}$$

where $\bar{K}_{C_j}$ and L_{C_i,C_j} are defined by:

$$\bar{K}_{C_j}=K_{C_j}\left(I_{C_j,C_j}-\frac{1}{n_j}1_{C_j,C_j}\right) \tag{2.67}$$

$$L_{C_i,C_j}=\left(\frac{1}{n_i}K_{C_i}1_{C_i}-\frac{1}{n_j}K_{C_j}1_{C_j}\right). \tag{2.68}$$

For a given sample $\vec{z}$, the corresponding feature $U_\phi^T\phi\left(\vec{z}\right)$ in the higher dimensional space is given by:

$$U_\phi^T\phi\left(\vec{z}\right)=\Lambda_{n\times H'}^T\left[\phi\left(\vec{x}_{i;j}\right)\right]_{H\times n}^T\phi\left(\vec{z}\right)=\Lambda_{n\times H'}^T\left[k\left(\vec{x}_{i;j},\vec{z}\right)\right]_{H\times n}^T, \tag{2.69}$$

where $k\left(\vec{x}_{i;j},\vec{z}\right)$ is the kernel function with entries $\vec{x}_{i;j}$ and $\vec{z}$. There are many typical kernel functions that could be used here, e.g., the Gaussian radial basis kernels ($k\left(\vec{x},\vec{x}'\right)=\exp\left(-\left\|\vec{x}-\vec{x}'\right\|^2/\left(2\sigma^2\right)\right)$), the homogeneous polynomial kernels ($k\left(\vec{x},\vec{x}'\right)=\left\langle\vec{x},\vec{x}'\right\rangle^d$), the inhomogeneous polynomial kernels ($k\left(\vec{x},\vec{x}'\right)=\left(\left\langle\vec{x},\vec{x}'\right\rangle+c\right)^d$), and the sigmoid kernels ($k\left(\vec{x},\vec{x}'\right)=\tanh\left(-\kappa\left\langle\vec{x},\vec{x}'\right\rangle+\vartheta\right)$). The homogeneous polynomial kernels and the Gaussian radial basis kernels are widely used in pattern classification. The homogeneous polynomial kernels are invariant under orthogonal transformations. Liu [87] empiricially demonstrated that the fractional powers in the homogeneous polynomial kernels perform better than integral powers. The radial basis kernels can be written as $k\left(\vec{x},\vec{x}'\right)=f\left(d\left(\vec{x},\vec{x}'\right)\right)$, where f is a function on R^+ and d is a metric function. In Gaussian radial basis kenrles, f is an exponential function and d is the Euclidean metric. In different applications, we can choose different f and d through the cross–validation to achieve reasonable performances.

Theorem 2.1: MGMKLD followed by KPCA is equal to KMGMKLD.

Proof: See Appendix.

Theorem 2.2: M–MGMKLD followed by KPCA is equal to the multimodal extension of KMGMKLD, or briefly M–KMGMKLD.

Proof. The result follows from the Theorem 2.1. ■

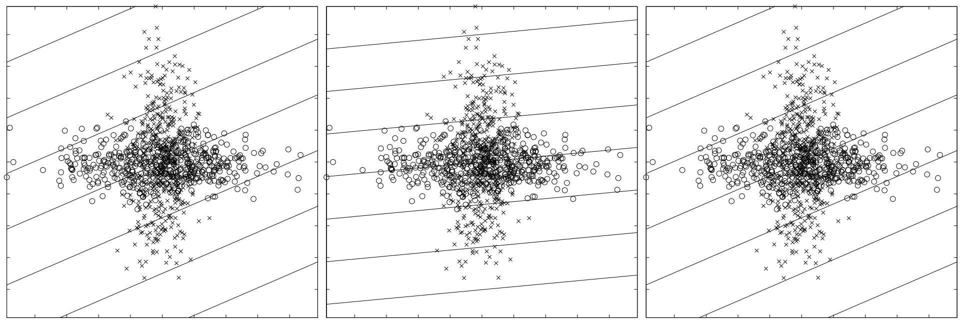
In the Theorem 2.1, we show that kernel principal component analysis (KPCA) following MGMKLD yields kernel MGMKLD. Based on this theorem, KMGMKLD is equivalent to first preprocessing the data using KPCA and then applying orthogonal MGMKLD to the preprocessed data.

Comparison using Synthetic Data

In this Section, we compare MGMKLD with the previous subspace selection methods, which are LDA [39], heteroscedastic discriminant analysis (HDA) [61], approximate pairwise accuracy criteria (aPAC) [98], weighted LDA (WLDA), fractional–step LDA (FS–LDA) [95], heteroscedastic extenstion LDA (HLDA) [99], oriented discriminant analysis (ODA) [28], and multimodal oriented discriminant analysis (MODA) [28]. WLDA is similar to aPAC, but the weighting function is d^{-8}. In FS–LDA, the weighting function is d^{-8} and the number of the fractional step is 30. We denote the proposed method as MGMKLD(η), where η is the combination factor in (2.48). We do not compare the proposed methods with the nonparametric based methods [2][16][94][90], because our framework is parametric.

Heteroscedastic Problem

To examine the classification ability of these subspace selection methods for solving the heteroscedastic problem [99], we generate two classes such that each class has 500 samples, drawn from a Gaussian distribution. The two Gaussian distributions have identical mean values but different covariances. As shown in Figure 2.4, LDA, aPAC, WLDA, and FS–LDA separate classes without taking the differences of class covariances into account. All other methods achieve better separation of the two classes as shown in Figure 2.4, because they consider the differences both in class means and in class covariances.



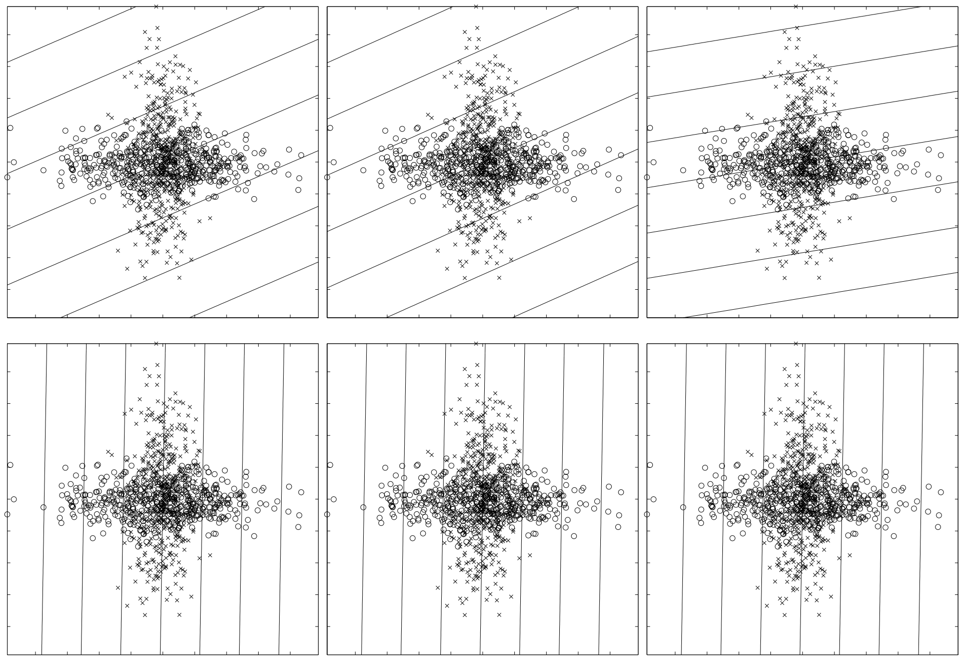


Figure 2.4. Heteroscedastic example: in this figure, from left to right, from top to bottom, there are nine subfigures showing the projection directions (indicated by lines in each subfigure) obtained using LDA, HDA, aPAC, WLDA, FS–LDA, HLDA, ODA, MGMKLD(0), and MGMKLD(1). In this experiment, the linear classifier $\left(x-\hat{m}_i\right)^T \hat{\Sigma}_i^{-1}\left(x-\hat{m}_i\right)+\ln\left|\hat{\Sigma}_i\right|$ is applied as the classification method after a subspace selection procedure, where $\hat{m}_i$ and $\hat{\Sigma}_i$ are the estimated i^{th} class mean and covariance matrix in the lower dimensional space, respectively. The training classification errors of these methods are 0.3410, 0.2790, 0.3410, 0.3410, 0.3430, 0.2880, 0.2390, 0.2390, and 0.2390, respectively.

Based on the same data, we demonstrate that the geometric mean of normalized divergences is not sufficient for subspace selection. On setting $\eta = 2$, MGMKLD reduces to the maximization of the geometric mean of normalized KL divergences. This result is based on the description of (2.48). The left subfigure in Figure 2.5 shows the projection direction (indicated by lines in each subfigure) from the two dimensional space to a one dimensional space found by maximizing the geometric mean of normalized KL divergences. The projection direction merges the two classes. The right subfigure in Figure 2.5 shows the geometric mean of normalized KL divergences in the one dimensional subspace during training. In this experiment, we observed: 1) KL divergences between the class 1 and class 2 is 1.1451 and normalized KL divergences is 0.5091 at the 1000th training iteration; and 2) the KL divergence between the class 2 and class 1 is 1.1043 and the normalized KL divergence is 0.4909 at the 1000th training iteration. The right subfigure shows normalized KL divergences are maximized finally, but the left subfigure shows the projection direction is not suitable for classification. The suitable projection direction for classification can be found in the bottom–left subfigure in Figure 2.4.

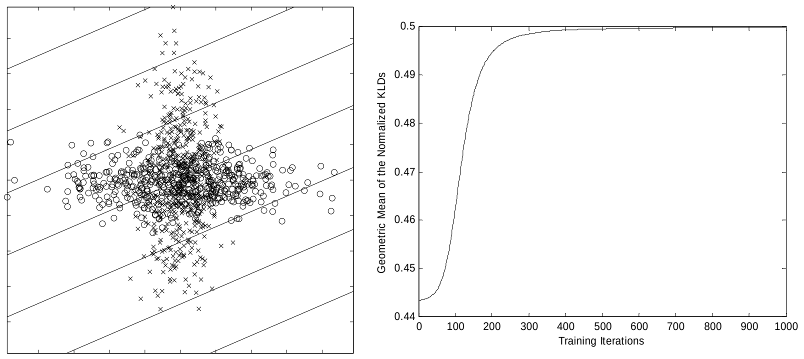


Figure 2.5. The maximization of the geometric mean of the normalized divergences is not sufficient for subspace selection.

Multimodal Problem

In many applications it is useful to model the distribution of a class by a GMM, because samples in the class may be drawn from a non–Gaussian distribution. To demonstrate the classification ability of M–MGMKLD, we generated two classes; each class has two subclusters; and samples in each subcluster are drawn from a Gaussian distribution. Figure 2.6 shows the subspaces selected by different methods. In this case, LDA, WLDA, FS–LDA, and aPAC do not select good subspaces for classification. However, the multimodal extensions of ODA and MGMKLD find suitable subspaces. Furthermore, although HDA and HLDA do not take account of multimodal classes, they each select a suitable subspace. This is because the two classes have different class covariance matrices when each class is modelled by a single Gaussian distribution. For complex cases, e.g., when each class consists of more than 3 subclusters, HDA and HLDA fail to find good subspaces for classification.

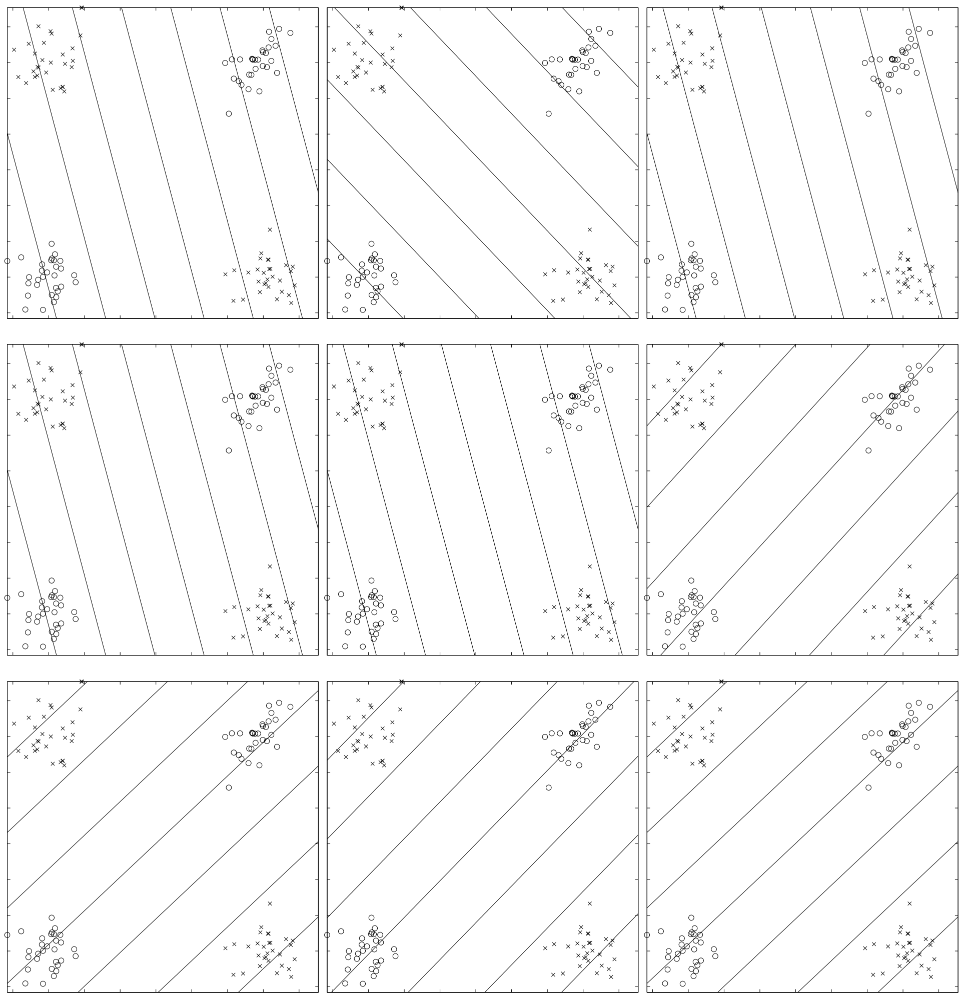


Figure 2.6. Multimodal problem: in this figure, from left to right, from top to bottom, there are nine subfigures, which show the projection directions (indicated by lines in each subfigure) by using LDA, HDA, aPAC, FS–LDA(3), FS–LDA(8), HLDA, MODA, M–MGMKLD(0), and M–MGMKLD(1). In this experiment, the linear classifier $\left(x-\hat{m}_i\right)^T \hat{\Sigma}_i^{-1}\left(x-\hat{m}_i\right)+\ln\left|\hat{\Sigma}_i\right|$ is applied as the classification method after a subspace selection procedure, where $\hat{m}_i$ and $\hat{\Sigma}_i$ are the estimated i^{th} class mean and covariance matrix in the lower dimensional space, respectively. The training classification errors of these methods are 0.0917, 0.0167, 0.0917, 0.0917, 0.0917, 0.0167, 0.0083, 0.0083, and 0.0083, respectively.

Class Separation Problem

The most prominent advantage of MGMKLD is it can significantly reduce the classification errors caused by very strong effects of large divergences between certain classes. To demonstrate this point, we generate three classes for which the samples in each class are drawn from Gaussian distributions. The KL divergence between two of these classes is small and the KL divergences between the third class and the two classes are large, i.e., two classes are close together and the third is further away.

In this case, MGMKLD(0) also performs well, but not as well as MGMKLD(5). In Figure 2.7, it is shown that MGMKLD(5) separates all classes. Furthermore, in MGMKLD, we achieve the same results when setting η, defined in (2.48), equal to 1,2,3,4, and 5. However, LDA, HLDA, and ODA do not give good results. The aPCA and FS–LDA algorithms are better than LDA but neither of them gives the best projection direction. The result obtained from aPCA is better than that obtained from WLDA, because aPAC uses a better weighting strategy than WLDA.

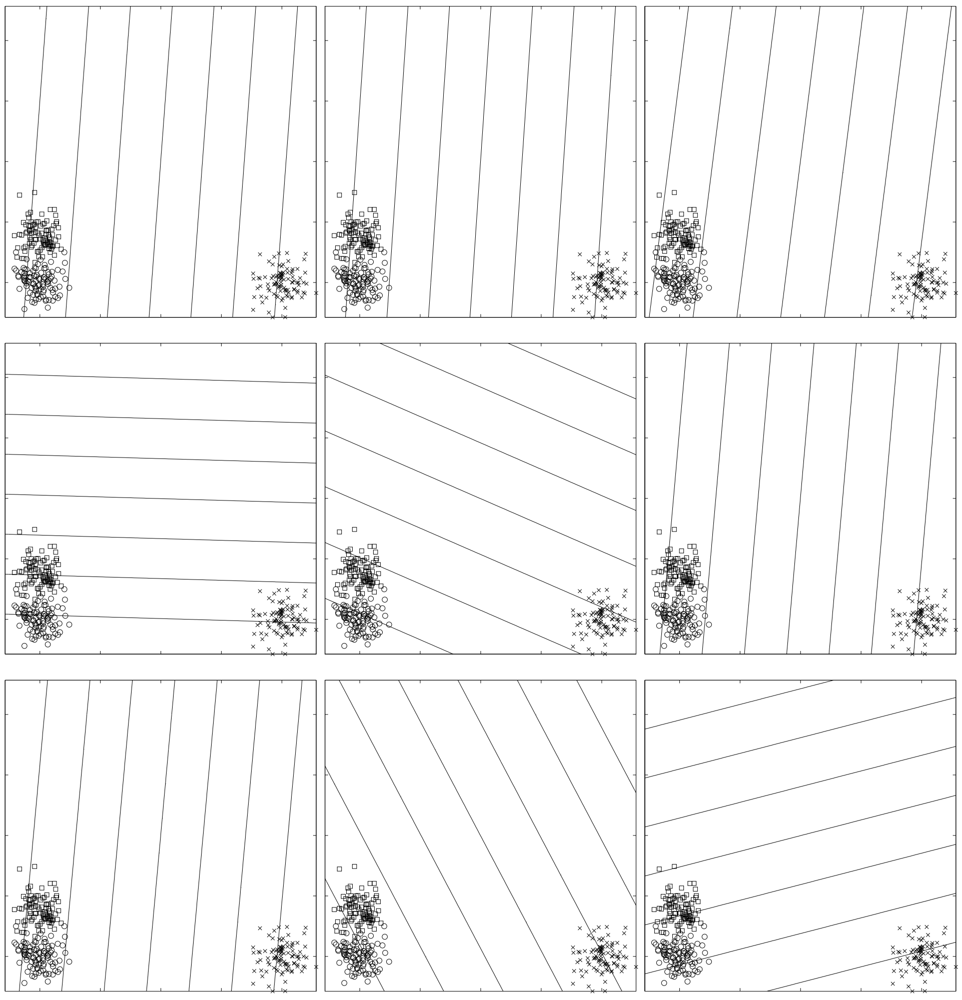


Figure 2.7. Class separation problem: in this figure, from left to right, from top to bottom, there are nine subfigures to describe the projection directions (indicated by lines in each subfigure) by using LDA, HDA, aPAC, WLDA, FS–LDA, HLDA, ODA, MGMKLD(0), and MGMKLD(5). In this experiment, the linear classifier $\left(x-\hat{m}_i\right)^T \hat{\Sigma}_i^{-1}\left(x-\hat{m}_i\right)+\ln\left|\hat{\Sigma}_i\right|$ is applied as the classification method after a subspace selection procedure, where $\hat{m}_i$ and $\hat{\Sigma}_i$ are the estimated i^{th} class mean and covariance matrix in the lower dimensional space, respectively. The training classification errors of these methods are 0.3100, 0.3033, 0.2900, 0.3033, 0.0567, 0.3100, 0.3100, 0.1167, and 0.0200, respectively. MGMKLD(5) finds the best projection direction for classification.

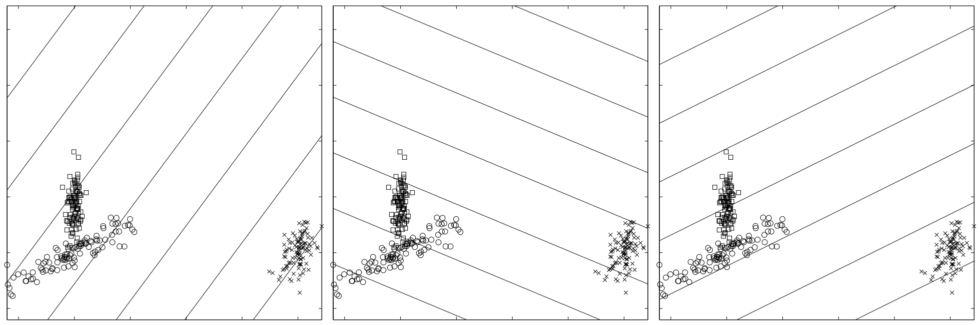


Figure 2.8. Class separation problem: in this figure, from left to right, there are three subfigures to describe the projection directions (indicated by lines in each subfigure) by using aPAC, FS–LDA, and MGMKLD(5). In this experiment, the linear classifier $\left(x-\hat{m}_i\right)^T \hat{\Sigma}_i^{-1}\left(x-\hat{m}_i\right)+\ln\left|\hat{\Sigma}_i\right|$ is applied as the classification method after a subspace selection procedure, where $\hat{m}_i$ and $\hat{\Sigma}_i$ are the estimated i^{th} class mean and covariance matrix in the lower dimensional space, respectively. The training classification errors of these methods are 0.1200, 0.1733, and 0.0267, respectively. MGMKLD(5) finds the best projection direction for classification.

In Figure 2.7, different Gaussians have identical covariances. In this case, FS–LDA works better than aPAC. In Figure 2.8, different Gaussians have different covariances. In this case, aPAC works better than FS–LDA. In both cases, the proposed MGMKLD (5) achieves the best performance.

Statistical Experiments

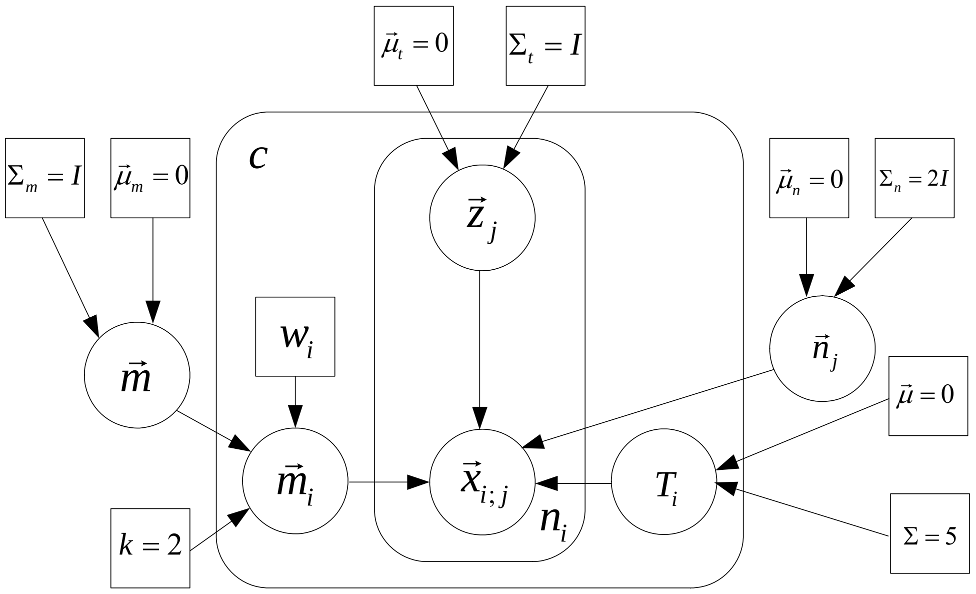


Figure 2.9. Data Generation Model. In this model, $w_1 = 4$, $w_2 = 0$, $w_3 = -4$, $w_4 = 4$, $w_5 = 4$, $k = 2$, $\vec{m}_1 = (k\vec{m} + w_1)1_{20}^T$, $\vec{m}_2 = 0_{20}^T$, $\vec{m}_3 = (k\vec{m} + w_3)\left[0_{10}, 1_{10}\right]^T$, $\vec{m}_4 = (k\vec{m} + w_4)\left[1_{10}, 0_{10}\right]^T$, and $\vec{m}_5 = (k\vec{m} + w_5)\left[1_5, 0_5, 1_5, 0_5\right]^T$.

In this Section, we utilize a synthetic data model, which is a generalization of the data generation model used by De la Torre and Kanade [28], to evaluate MGMKLD in terms of accuracy and robustness. The accuracy is measured by averaged classification errors and the robustness is measured by standard deviation of the classification errors. In this experiment, the linear classifier [30] and the nearest neighbour rule[9] [105] are applied for classification after a subspace selection procedure. In this data generation model, there are five classes, which are represented by the symbols ○, ×, +, □, and ◇, as shown in Figure 2.11 (on page 62). In our experiments, for the training/testing set, the data generator gives 200 samples for each of the five classes (therefore, 1,000 samples in total). Moreover, the samples in each class are obtained from a single Gaussian. Each Gaussian density is a linear transformed "standard Gaussian distribution",

[9] The nearest neighbor rule classifies a sample $\vec{x}$ to the class C, when $\vec{x}'$ is the nearest neighbor to $\vec{x}$ under the Euclidean metric and $\vec{x}'$ belongs to the class C.

i.e., $N(0,I)$. The linear transformations are defined by $\vec{x}_{i;j} = T_i\vec{z}_j + \vec{m}_i + \vec{n}_j$, where $\vec{x}_{i;j} \in R^{20}$, $T_i \in R^{20\times 7}$, $\vec{z}_j \sim N(0,I) \in R^7$, $\vec{n}_j \sim N(0,2I) \in R^{20}$, i denotes the *i*th class, *j* denotes the *j*th sample in this class, and $\vec{m}_i$ is the mean value of the normal distribution for the *i*th class. The $\vec{m}_i$ are assigned with the following values: $\vec{m}_1 = (2N(0,1)+4)1_{20}^T$, $\vec{m}_2 = 0_{20}^T$, $\vec{m}_3 = (2N(0,1)-4)[0_{10},1_{10}]^T$, $\vec{m}_4 = (2N(0,1)+4)[1_{10},0_{10}]^T$, and $\vec{m}_5 = (2N(0,1)+4)[1_5,0_5,1_5,0_5]^T$, where 1_{20} is a row vector in R^{20} and all entries in 1_{20} are 1. The notations of 1_{10}, 1_5, 0_{20}, 0_{10}, and 0_5 have the similar meanings as 1_{20}. The projection matrix T_i is a random matrix, in which the elements are sampled independently from $N(0,5)$, where 5 is the variance. The data generation model is shown in Figure 2.9. Based on this data generation model, 800 groups (each group consists of training and testing samples) of synthetic data are generated.

For comparison, the subspace selection methods, e.g., MGMKLD, are first utilized to select a given number of features. Then the nearest neighbour rule and the linear classifier $\left(\vec{x}-\hat{\vec{m}}_i\right)^T \hat{\Sigma}_i^{-1}\left(\vec{x}-\hat{\vec{m}}_i\right)+\ln\left|\hat{\Sigma}_i\right|$ are used as the classification methods after a subspace selection procedure, where $\hat{\vec{m}}_i$ and $\hat{\Sigma}_i$ are the estimated *i*th class mean and covariance matrix in the lower dimensional space, respectively. In this Section, the baseline algorithms are LDA, HDA, aPAC, WLDA, FS–LDA, HLDA, and ODA.

Performance Evaluation

We conducted the designed experiments 800 times based on randomly generated data sets. The experimental results are reported in Table 2.7 – Table 2.10. Table 2.7 and Table 2.9 show averaged classification errors of LDA, HDA, aPAC, WLDA, FS–LDA, HLDA, ODA, MGMKLD(0), and MGMKLD(2*c*). Table 2.7 and Table 2.8 show the results of the nearest neighbour rule and Table 2.9 and Table 2.10 show the results of the linear classifier. For the 800 experiments, statistical experimental results are shown in Table 2.7 and Table 2.9, where arithmetic mean values are computed on different feature dimensions from 1 to 6 (by column). Correspondingly, standard deviations under each condition, which measure the robustness of the classifiers, are given in Table 2.8 and Table 2.10. We emphasize that we have twenty feature dimensions for each sample and all samples are divided into one of the five classes, therefore, the maximal feature number for LDA, HDA, WLDA, aPAC, and FS–LDA is 5–1=4; in contrast, HLDA, ODA, and MGMKLD can extract more features than LDA and HDA. From Table 2.7 – Table 2.10, it can be concluded that MGMKLD outperforms LDA, HDA, aPAC, WLDA, FS–LDA, HLDA, and ODA, consistently. Finally, the linear classifier distance outperforms the nearest neighbour rule when samples are sampled from Gaussian distributions.

Table 2.7: *Averaged classification errors* (the mean for 800 experiments) of LDA, HDA, aPAC, WLDA, FS–LDA, HLDA, ODA, MGMKLD(0), and MGMKLD(2*c*). (The nearest neighbour rule)

Basis	1	2	3	4	5	6
LDA	0.2968	0.1552	0.1103	0.1504	—	—
HDA	0.3087	0.1850	0.1642	0.1807	—	—
aPAC	0.3206	0.1469	0.1088	0.1324	—	—
WLDA	0.4320	0.1930	0.1126	0.1092	—	—
FS–LDA	0.3098	0.1490	0.1108	0.1104	—	—
HLDA	0.2982	0.1561	0.1073	0.1050	0.1043	0.1043
ODA	0.3029	0.1706	0.1370	0.1266	0.1219	0.1206
MGMKLD(0)	0.2548	0.1397	0.1054	0.1030	0.1024	0.1018
MGMKLD(2*c*)	**0.2430**	**0.1310**	**0.1039**	**0.1018**	**0.1010**	**0.1006**

Table 2.8: S*tandard deviations of classification errors* for 800 experiments of LDA, HDA, aPAC, WLDA, FS–LDA, HLDA, ODA, MGMKLD(0), and MGMKLD(2*c*). (The nearest neighbour rule)

Basis	1	2	3	4	5	6
LDA	0.1002	0.0995	0.0985	0.1077	—	—
HDA	0.1016	0.1040	0.1007	0.1055	—	—
aPAC	0.1270	0.1171	0.0979	0.0983	—	—
WLDA	0.1275	0.1239	0.1051	0.0995	—	—
FS–LDA	0.1475	0.1216	0.0990	0.0964	—	—
HLDA	0.1001	0.0992	0.0968	0.0942	0.0928	0.0920
ODA	0.1010	0.1037	0.1016	0.0992	0.0965	0.0957
MGMKLD(0)	0.1094	0.0985	0.0948	0.0919	0.0905	0.0893
MGMKLD(2*c*)	0.1130	0.0979	0.0947	0.0916	0.0902	0.0889

Table 2.9: *Averaged classification errors* (the mean for 800 experiments) of LDA, HDA, aPAC, WLDA, FS–LDA, HLDA, ODA, MGMKLD(0), and MGMKLD(2*c*). (The linear classifier $\left(\vec{x}-\hat{\vec{m}}_i\right)^T \hat{\Sigma}_i^{-1}\left(\vec{x}-\hat{\vec{m}}_i\right)+\ln\left|\hat{\Sigma}_i\right|$)

Basis	1	2	3	4	5	6
LDA	0.2455	0.1199	0.0811	0.0813	—	—
HDA	0.2516	0.1384	0.0971	0.1221	—	—
aPAC	0.2626	0.1134	0.0827	0.0813	—	—
WLDA	0.3716	0.1498	0.0863	0.0817	—	—
FS–LDA	0.3050	0.1363	0.0883	0.1090	—	—
HLDA	0.2456	0.1216	0.0821	0.0791	0.0764	0.0741
ODA	0.2500	0.1327	0.1037	0.0894	0.0829	0.0796
MGMKLD(0)	**0.2226**	0.1099	0.0815	0.0776	0.0751	0.0725
MGMKLD(2*c*)	0.2404	**0.1036**	**0.0806**	**0.0766**	**0.0742**	**0.0720**

Table 2.10: S*tandard deviations of classification errors* for 800 experiments of LDA, HDA, aPAC, WLDA, FS–LDA, HLDA, ODA, MGMKLD(0), and MGMKLD(2*c*). (The linear classifier $\left(\vec{x}-\hat{\vec{m}}_i\right)^T \hat{\Sigma}_i^{-1}\left(\vec{x}-\hat{\vec{m}}_i\right)+\ln\left|\hat{\Sigma}_i\right|$)

Basis	1	2	3	4	5	6
LDA	0.0932	0.0843	0.0792	0.0795	—	—
HDA	0.0919	0.0880	0.0886	0.1435	—	—
aPAC	0.1175	0.0987	0.0816	0.0795	—	—
WLDA	0.1231	0.1055	0.0870	0.0804	—	—
FS–LDA	0.1678	0.1387	0.0980	0.1272	—	—
HLDA	0.0919	0.0852	0.0799	0.0772	0.0745	0.0725
ODA	0.0923	0.0894	0.0879	0.0810	0.0766	0.0739
MGMKLD(0)	0.1033	0.0844	0.0788	0.0746	0.0720	0.0695
MGMKLD(2*c*)	0.1373	0.0848	0.0795	0.0743	0.0717	0.0692

Initial Values Issue

We generate a training set according to the synthetic data model described at the beginning of this Section. We randomly initialize parameters in MGMKLD(2*c*) to examine how different initial values affect the solution. Note that we omit lines 1, 2, 6, and 7 from the optimization procedure for MGMKLD given in Table 2.5, because these lines are used to set different initial values. We can see from Figure 2.10 that MGMKLD(2*c*) is insensitive to choices of initial values in 50 random experiments.

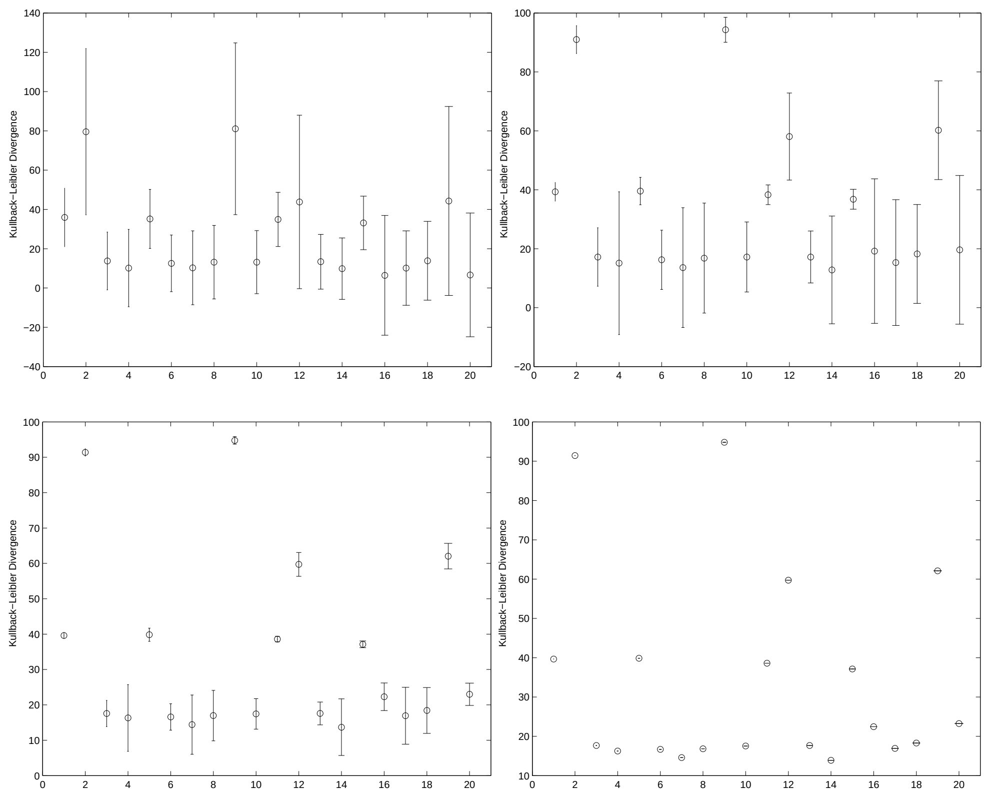
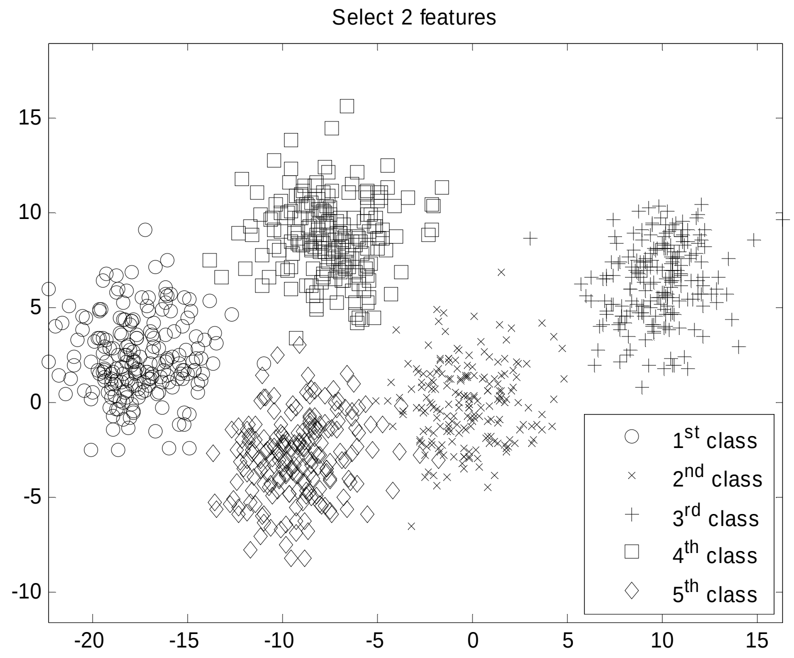


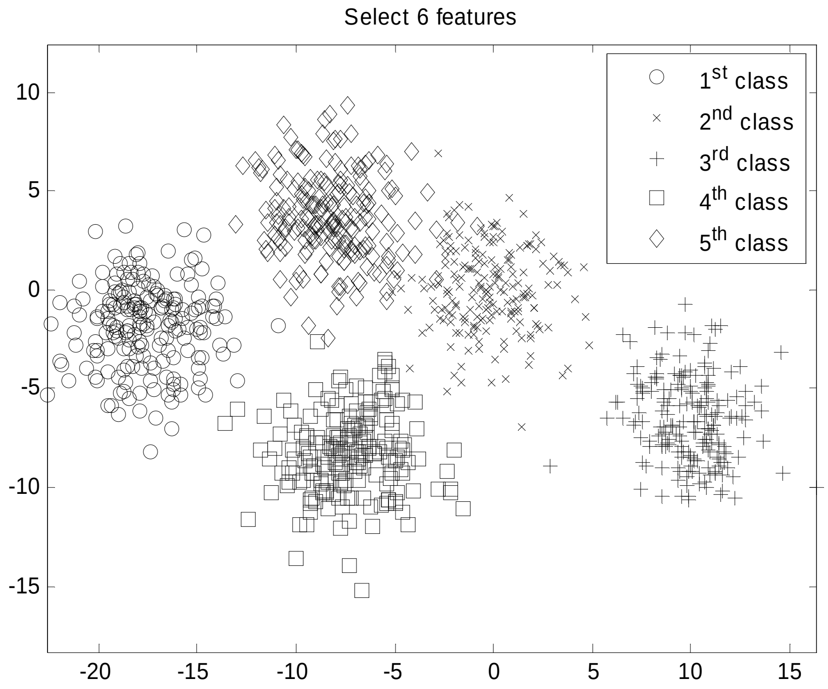
Figure 2.10. Initial values: From left to right, from top to bottom, the subfigures show the mean value and the corresponding standard deviation of the KL divergence between the class *i* and the class *j* of these 50 different initial values in the 10th (50th, 100th, and 1000th) training iterations. Because there are 5 classes in the training set, there are 20 KL divergences to examine. The circles in each subfigure show the mean values of the KL divergences for 50 different initial values. The error bars show the corresponding standard deviations. For better visualization, the scale for showing the standard deviations is 10 times larger than the vertical scale in each subfigure. The standard deviations of these 20 KL divergences approach 0 as the number of training iterations increases.

Nest Structure Property

Definition 2.4: Given a subspace selection method, $W_n = \left[\vec{w}_i\right]_{1\le i\le n}$ and $U_m = \left[\vec{u}_i\right]_{1\le i\le m}$ are projection matrices obtained from the same training data but with $m > n$, i.e., U_m contains more features than W_n, i.e., $m > n$. Let U_n be the first *n* columns of U_m. If U_n and W_n are identified to the same point on a

Grassmann manifold, we say the subspace selection method has the *nest structure* property. For example, PCA has the nest structure property.





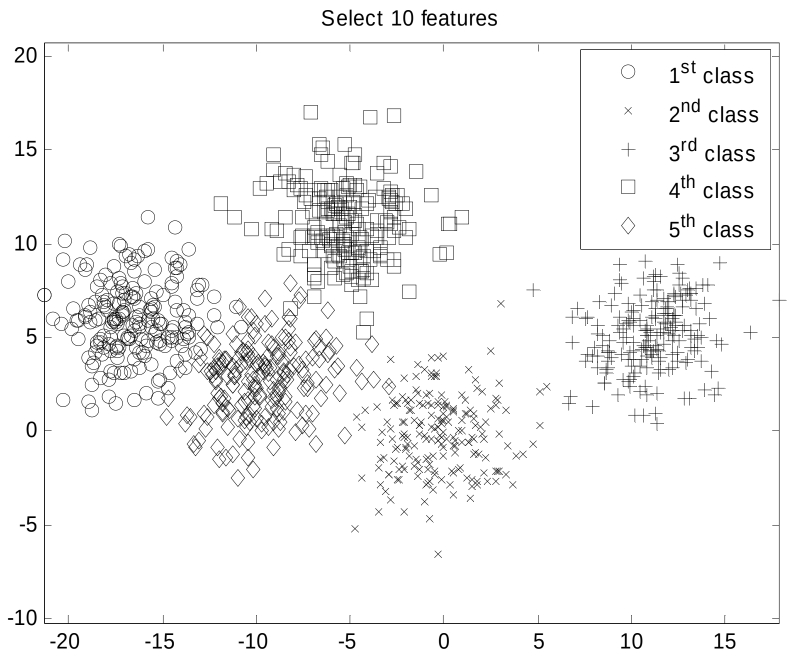


Figure 2.11. MGMKLD(2*c*) has no nest structure property.

A desirable subspace selection method should adapt to the selected dimensions. A subspace selection method of this type is not expected to have the nest structure property. For instance, in our experiment, we first extract two dimensional features from the original entire feature set (twenty features) based on MGMKLD(2*c*); with these two features, the profile of the five classes of samples is illustrated in the first subfigure of Figure 2.11. We then extract six features from all based on MGMKLD(2*c*) with the same training samples, but we only show the first two dimensions of the extracted six dimensional features in the second subfigure of Figure 2.11. The third subfigure shows the first two dimensions when ten features are extracted. Based on the figure, we can see that these first two features in the two cases are different. Therefore, MGMKLD(2*c*) has no nest structure.

Real–World Experiments

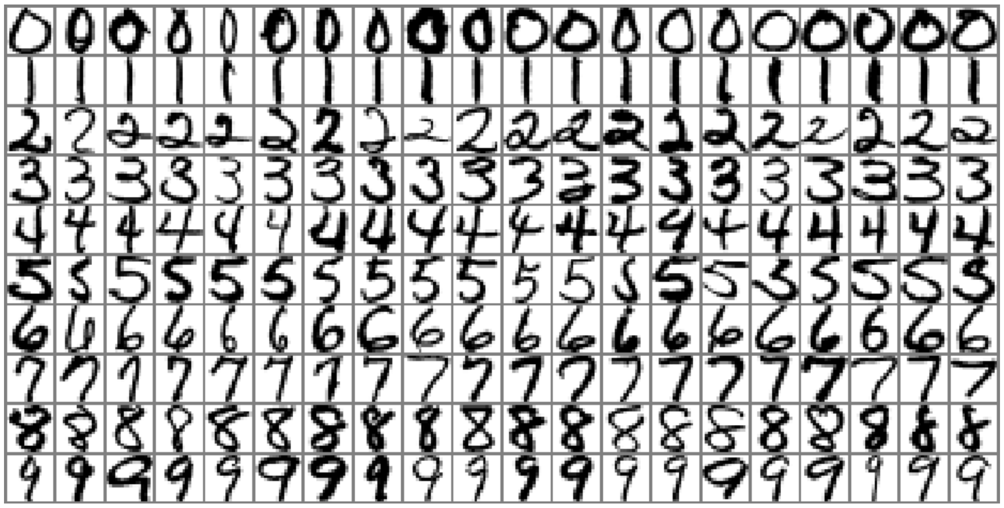


Figure 2.12. Samples in the USPS database [53].

In this Section, we report the experimental results of the proposed algorithms using a well known character recognition data set, the United States Postal Services (USPS) database [53], in which there are 9,298 handwriting character samples divided into ten classes. Twenty samples of each class are given in Figure 2.12 row by row. Each sample is a vector with 256 dimensions. The entire USPS data set is divided into two parts [53], a training set with 7,291 samples and a test set with 2,007 samples. In our experiments, we utilize the entire USPS database to evaluate performances of LDA, HDA, aPAC, WLDA, FS–LDA, HLDA, ODA, MODA, MGMKLD(0), MGMKLD(2c), and M–MGMKLD(2c).

Linear Method

We apply the algorithms to the USPS database. As illustrated in Table 2.11, when the top 3, 5, and 7 discriminative features are required, MGMKLD(2c) gives the best performance for all cases among all algorithms. When the top 9, 15, and 20 features are selected, M–MGMKLD(2c) consistently outperforms all other algorithms and gives comparable performance to MGMKLD(2c) in some cases. The error rate is only around half that of ODA and MODA. The use of GMM in M–MGMKLD(2c) is particularly advantageous for large data sets. Note that in the original 256 dimensional feature space, the number of the classes is ten, therefore, the maximum dimension of the extracted feature space is nine (10–1=9) for LDA, aPAC, WLDA, HDA, and FS–LDA, while for other listed algorithms, including our MGMKLD(2*c*) and M–MGMKLD(2*c*), more features can be selected. This is a minor advantage of our algorithms compared with the conventional LDA model.

Table 2.11. Performances (classification errors) of linear methods on the USPS database. (The nearest neighbour rule)

Basis	3	5	7	9	15	20
LDA	0.3827	0.1629	0.1186	0.1096	—	—
aPAC	0.3144	0.1679	0.2141	0.1106	—	—
HDA	0.3518	0.2516	0.1988	0.1734	—	—
WLDA	0.4310	0.2342	0.1415	0.1096	—	—
FS–LDA	0.3338	0.1794	0.1430	0.1131	—	—
HLDA	0.3518	0.2347	0.1779	0.1281	0.0947	0.0827
ODA	0.3983	0.2617	0.1636	0.1162	0.1060	0.0970
MODA	0.3950	0.2850	0.1576	0.1032	0.1027	0.0937
MGMKLD(0)	0.2875	0.1574	0.1116	0.0987	0.0598	0.0583
MGMKLD(2*c*)	**0.2720**	**0.1375**	**0.1105**	0.0867	0.0578	0.0562
M–MGMKLD(2*c*)	0.3179	0.1414	0.1106	**0.0822**	**0.0569**	**0.0553**

Kernel Method

We compare KMGMKLD(2*c*) and M–KMGMKLD(2*c*) with the Kernel LDA (KDA) [104] in Table 2.12. Firstly, the results show that all three kernel algorithms perform better than the linear algorithms. Secondly, the improvements for LDA, MGMKLD(2*c*), and M–MGMKLD(2*c*) are 13.22%, 8.42%, and 23.11%, respectively. Herein, nine dimensions are selected to construct the kernel feature space. The kernel $k\left(\vec{x},\vec{x}'\right)=\left\langle \vec{x},\vec{x}'\right\rangle^2$ is chosen.

Table 2.12. Performances (error rates) of kernel methods on the USPS database. A nine dimensional feature space is selected for each algorithm. (The nearest neighbour rule)

Basis	KDA	KMGMKLD(2*c*)	M–KMGMKLD(2*c*)
Error	0.0951	0.0794	0.0632

Summary

General averaged divergences analysis is proposed to enhance the conventional Fisher–Rao linear discriminant analysis (LDA). LDA is one of the most important subspace methods in pattern classification research and applications; however, it has a tendency to merge together nearby classes when the features are projected to a lower dimensional space. To reduce this merging, new criteria for subspace selection are chosen. The new criteria are based on the geometric mean of the divergences between different pairs of classes. Three new criteria are defined, namely, 1) maximization of the geometric mean of the divergences; 2) maximization of the geometric mean of normalized divergences; and 3) maximization of the geometric mean of all divergences. The third criterion is a combination of the first two criteria. Then, the multimodal extension and the kernel extension of the maximization of the geometric mean of all divergences are introduced as well. The divergence can be any Bregman divergence. In our experiments we use the Kullback–Leibler divergence, which is a special case of the Bregman divergence.

The new subspace selection methods are tested experimentally using synthetic data and handwriting data from the USPS database [53]. The experiments show that the third criterion, named the maximization of the geometric mean of all KL divergences is more effective than LDA and its representatives. In the future, we will utilize this method for other pattern classification tasks, for example biometrics and bioinformatics.

3. Discriminative Multilinear Subspace Method

In computer vision research, many objects are naturally represented by multidimensional arrays, i.e., tensors [75], such as the gray face image shown in Figure 3.1 in face recognition [34][168], the color image shown in Figure 3.2 in scene image classification [156], and the video shot shown in Figure 3.3 in motion categorization [43]. However, in current research, the original tensors (images and videos) are always scanned into vectors, thus discarding a great deal of useful structural information, which is helpful to reduce the small sample size (SSS) problem in subspace selection methods, e.g., linear discriminant analysis (LDA).

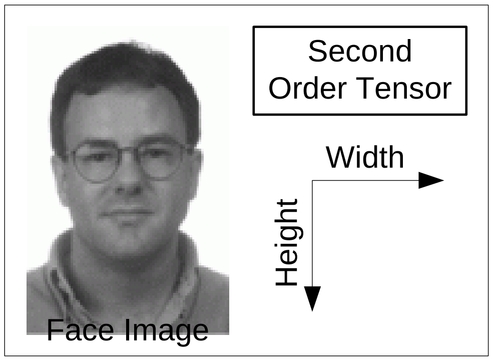


Figure 3.1. A gray level face image is a second order tensor, i.e., a matrix. Two indices are required for pixel locations. The face image comes from http://www.merl.com/projects/images/face–rec.gif.

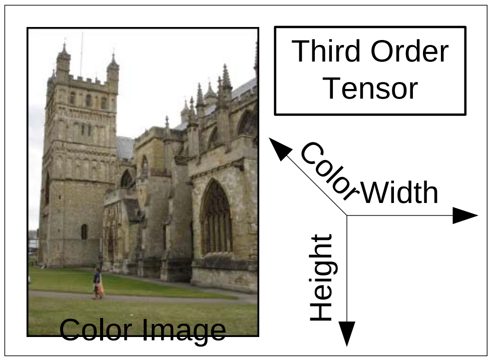


Figure 3.2. A color image is a third order tensor, which is also a data cuboid, because three indices are required to locate elements. Two indices are used for pixel locations and one index is used to local the color information (e.g., R, G, and B).

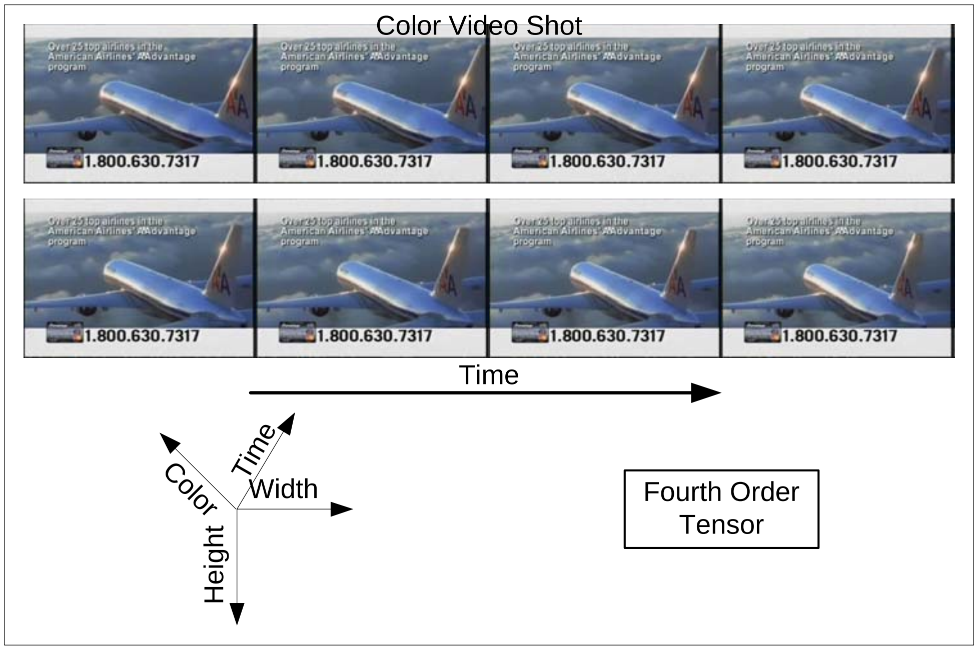


Figure 3.3. A color video shot is a fourth order tensor. Four indices are used to locate elements. Two indices are used for pixel locations; one index is used to locate the color information; and the other index is used for time. The video shot comes from http://www–nlpir.nist.gov/projects/trecvid/.

To utilize the structure information, many dimension reduction algorithms [75] [132][162][171] based on the multilinear subspace method (MLSM) have been developed for data representation [75][162][171][132], pattern classification [162][171][132], and network abnormal detection [136]. MLSM finds a sequence of linear transformation matrices $U_i \in R^{L_i \times L_i'}$ ($L_i' < L_i$, $1 \le i \le M$) to transform a large tensor $\mathbf{X} \in R^{L_1 \times L_2 \times \cdots L_M}$ to a smaller tensor $\mathbf{Y} \in R^{L_1' \times L_2' \times \cdots L_M'}$. For example, if we have a second order tensor $\mathbf{X} \in R^{L_1 \times L_2}$ with large L_1 and L_2, in MLSM we need to find two linear transformation matrices $U_1 \in R^{L_1 \times L_1'}$ ($L_1' < L_1$) and $U_2 \in R^{L_2 \times L_2'}$ ($L_2' < L_2$) to transform $\mathbf{X}$ according to $\mathbf{Y} = U_1^T \mathbf{X} U_2$. After the transformation, the dimension is reduced from $L_1 \times L_2$ to $L_1' \times L_2'$, i.e., $\mathbf{Y} \in R^{L_1' \times L_2'}$. Similar to LSM, MLSM includes a large number of methodologies varying from reconstructive models to discriminative models based on different criteria. The reconstructive models, e.g., the general tensor analysis (GTA) [75][162][171] and the tensor rank one analysis (TR1A) [132], are utilized to generate low dimensional representations which preserve the original information as much as possible. The discriminative models, e.g., the two dimensional linear discriminant analysis (2DLDA) [173] and the general tensor discriminant analysis (GTDA) [144][147], find low dimensional representations which preserve as much as possible of the information required for reliable classification.

In this Chapter, we show that the SSS problem is reduced if the tensor structure of data is retained. We also demonstrate that MLSM is a constrained version of the linear subspace method (LSM), i.e., MLSM is equivalent to LSM combined with constraints.

In this Chapter, we mainly focus on the discriminative MLSM, especially 2DLDA, which is a two dimensional extension of LDA. That is 2DLDA accepts matrix type data as input while LDA accepts vector type data as input. The effectiveness and the efficiency of 2DLDA for dimension reduction have been demonstrated in face recognition. However, 2DLDA fails to converge during the training stage. This nonconvergence has the following disadvantages: 1) it is difficult to determine when to stop the training stage; and 2) different numbers of training iterations lead to different recognition results. To solve the nonconvergence problem in 2DLDA, we develop GTDA, which is an M dimensional extension of LDA. That is GTDA accepts general tensors (e.g., vectors, matrices, data cuboids, and high dimensional arrays) as input. GTDA has the following properties: 1) reduction of the SSS problem for subsequent classification, e.g., by LDA; 2) preservation of the discriminative information in training tensors; 3) provision with a converged iterative optimization algorithm, which obtains a stable solution for GTDA; and 4) acceptance of general tensors as input.

The organization of this Chapter is as follows. In §147, the mathematical foundation, tensor algebra, of this thesis is briefly introduced. In §147, we describe the relationship between the LSM algorithms and the MLSM algorithms. In §147, the tensor rank one analysis based on the best rank one approximation is reviewed. In §147, the general tensor analysis based on the best rank $\left(R_1, R_2, \cdots R_M\right)$ approximation is reviewed. In §147, we analyze the nonconvergence issue of the 2DLDA for discriminative subspace selection. In §147, we develop GTDA and GTDA's manifold learning extension is given in §147. To examine the effectiveness of GTDA, it is applied for human gait recognition and the performance evaluation is given in §147. Finally, the summary of this Chapter is given in §147 and the relevant deductions and proofs in this Chapter are given in §147.

Tensor Algebra

This Section contains the fundamental materials on tensor algebra [75], which are relevant to this thesis. Tensors are arrays of numbers which transform in certain ways under different coordinate transformations. The order of a tensor $\mathbf{X} \in R^{L_1 \times L_2 \times \cdots \times L_M}$, represented by a multi–dimensional array of real numbers, is M. An element of $\mathbf{X}$ is denoted as $\mathbf{X}_{l_1, l_2, \ldots, l_M}$, where $1 \le l_i \le L_i$ and $1 \le i \le M$. The ith dimension (or mode) of $\mathbf{X}$ is of size L_i. A scalar is a zero–th order tensor; a vector is a first order tensor; and a matrix is a second order tensor. The structure of a third order tensor is shown in Figure 3.4. In the tensor terminology, we have the following definitions.

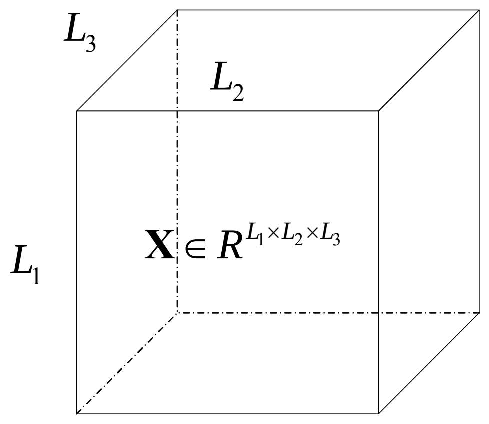


Figure 3.4. A third order tensor $\mathbf{X} \in R^{L_1 \times L_2 \times L_3}$.

Definition 3.1 (Tensor Product or Outer Product) The tensor product $\mathbf{X} \otimes \mathbf{Y}$ of a tensor $\mathbf{X} \in R^{L_1 \times L_2 \times \cdots \times L_M}$ and another tensor $\mathbf{Y} \in R^{L'_1 \times L'_2 \times \cdots \times L'_{M'}}$ is defined by

$$\left(\mathbf{X} \otimes \mathbf{Y}\right)_{l_1 \times l_2 \times \ldots \times l_M \times l'_1 \times l'_2 \times \ldots \times l'_{M'}} = \mathbf{X}_{l_1 \times l_2 \times \ldots \times l_M} \mathbf{Y}_{l'_1 \times l'_2 \times \ldots \times l'_{M'}}, \tag{3.01}$$

for all index values.

For example, the tensor product of two vectors $\vec{x}_1 \in R^{L_1}$ and $\vec{x}_2 \in R^{L_2}$ is a matrix $X \in R^{L_1 \times L_2}$, i.e., $X = \vec{x}_1 \otimes \vec{x}_2 = \vec{x}_1 \vec{x}_2^T$.

Definition 3.2 (Mode–*d* Matricizing or Matrix Unfolding) The mode–*d* matricizing or matrix unfolding of an *M*–th order tensor $\mathbf{X} \in R^{L_1 \times L_2 \times \cdots \times L_M}$ is the set of vectors in R^{L_d} obtained by keeping the index i_d fixed and varying the other indices. Therefore, the mode–*d* matricizing or matrix unfolding of an *M*–th order tensor is a matrix $X_{(d)} \in R^{L_d \times \bar{L}_d}$, where $\bar{L}_d = \left(\prod_{i \neq d} L_i\right)$. We denote the mode–*d* matricizing of $\mathbf{X}$ as $\mathrm{mat}_d(\mathbf{X})$ or briefly $X_{(d)}$. Figure 3.5 shows the mode–1, mode–2, and mode–3 matricizing of a third order tensor $\mathbf{X} \in R^{3\times 2\times 2}$.

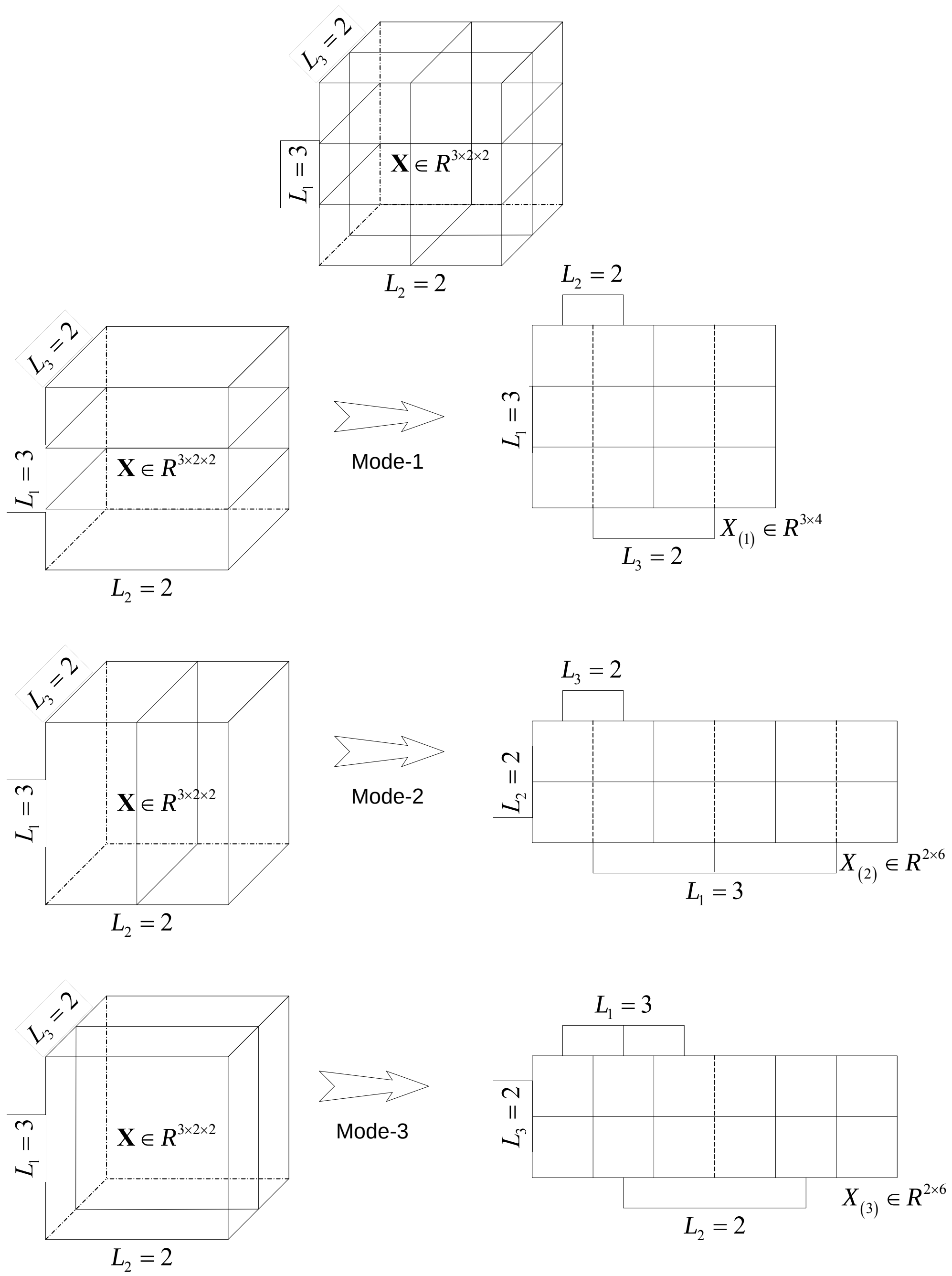

$L_3 = 2$
$L_1 = 3$
$\mathbf{X} \in R^{3\times 2\times 2}$
$L_2 = 2$
Mode-1
$X_{(1)} \in R^{3\times 4}$
Mode-2
$X_{(2)} \in R^{2\times 6}$
$L_1 = 3$
Mode-3
$X_{(3)} \in R^{2\times 6}$

Figure 3.5. The mode–1, mode–2, and mode–3 matricizing of a third order tensor $\mathbf{X} \in R^{3\times 2\times 2}$.

Definition 3.3 (Tensor Contraction) The contraction of a tensor is obtained by equating two indices and summing over all values of repeated indices. Contraction reduces the tensor order by 2.

A suitable notation for contraction is the Einstein's summation convention[10]. For example, the tensor product of two vectors $\vec{x}, \vec{y} \in R^N$ is $Z = \vec{x} \otimes \vec{y}$; and the contraction of Z is $Z_{ii} = \vec{x} \cdot \vec{y} = \vec{x}^T \vec{y}$, where the repeated indices imply summation. The value of Z_{ii} is the inner product of $\vec{x}$ and $\vec{y}$. In general, for tensors $\mathbf{X} \in R^{L_1 \times \cdots \times L_M \times L_1' \times \cdots \times L_{M'}'}$ and $\mathbf{Y} \in R^{L_1 \times \cdots \times L_M \times L_1'' \times \cdots \times L_{M''}''}$, the contraction on the tensor product $\mathbf{X} \otimes \mathbf{Y}$ is

$$\llbracket \mathbf{X} \otimes \mathbf{Y}; (1:M)(1:M) \rrbracket = \sum_{l_1=1}^{L_1} \cdots \sum_{l_M=1}^{L_M} (\mathbf{X})_{l_1 \times \cdots \times l_M \times l_1' \times \cdots \times l_{M'}'} (\mathbf{Y})_{l_1 \times \cdots \times l_M \times l_1'' \times \cdots \times l_{M''}''}, \tag{3.02}$$

In this thesis, when the convention is conducted on all indices except the i^{th} index on the tensor product of $\mathbf{X}$ and $\mathbf{Y}$ in $R^{L_1 \times L_2 \times \cdots \times L_M}$, we denote this procedure as

$$\begin{aligned} \llbracket \mathbf{X} \otimes \mathbf{Y}; (\bar{i})(\bar{i}) \rrbracket &= \llbracket \mathbf{X} \otimes \mathbf{Y}; (1:i-1, i+1:M)(1:i-1, i+1:M) \rrbracket \\ &= \sum_{l_1=1}^{L_1} \cdots \sum_{l_{i-1}=1}^{L_{i-1}} \sum_{l_{i+1}=1}^{L_{i+1}} \cdots \sum_{l_M=1}^{L_M} (\mathbf{X})_{l_1 \times \cdots \times l_{i-1} \times l_i \times l_{i+1} \times \cdots \times l_M} (\mathbf{Y})_{l_1 \times \cdots \times l_{i-1} \times l_i \times l_{i+1} \times \cdots \times l_M} \\ &= \operatorname{mat}_i(\mathbf{X}) \operatorname{mat}_i^T(\mathbf{Y}) = X_{(i)} Y_{(i)}^T, \end{aligned} \tag{3.03}$$

and $\llbracket \mathbf{X} \otimes \mathbf{Y}; (\bar{i})(\bar{i}) \rrbracket \in R^{L_i \times L_i}$.

Definition 3.4 (Mode–*d* product) The mode–*d* product $\mathbf{X} \times_d U$ of a tensor $\mathbf{X} \in R^{L_1 \times L_2 \times \ldots \times L_M}$ and a matrix $U \in R^{L'_d \times L_d}$ is an $L_1 \times L_2 \times \cdots \times L_{d-1} \times L'_d \times L_{d+1} \times \cdots \times L_M$ tensor defined by

$$\begin{aligned} (\mathbf{X} \times_d U)_{l_1 \times l_2 \times \cdots \times l_{d-1} \times l_d' \times l_{d+1} \times \cdots \times l_M} &= \sum_{l_d'} \left(\mathbf{X}_{l_1 \times l_2 \times \cdots \times l_{d-1} \times l_d \times l_{d+1} \times \cdots \times l_M} U_{l_d' \times l_d} \right) \\ &= \llbracket \mathbf{X} \otimes U; (d)(2) \rrbracket, \end{aligned} \tag{3.04}$$

for all index values. The mode–*d* product is a type of contraction.

Based on the definition of Mode–*d* product, we have

$$(\mathbf{X} \times_d U) \times_t V = (\mathbf{X} \times_t V) \times_d U \tag{3.05}$$

[10] "*When any two subscripts in a tensor expression are given the same symbol, it is implied that the convention is formed.*" ———A. Einstein, Die Grundlage der Allgemeinen Relativitatstheorie, Ann. Phys., 49:769, 1916.

where $\mathbf{X} \in R^{L_1 \times L_2 \times \cdots \times L_M}$, $U \in R^{L'_d \times L_d}$, and $V \in R^{L'_t \times L_t}$. Therefore, $(\mathbf{X} \times_d U) \times_t V$ can be simplified as $\mathbf{X} \times_d U \times_t V$.

Furthermore,

$$(\mathbf{X} \times_d U) \times_t V = \mathbf{X} \times_d (VU) \tag{3.06}$$

where $\mathbf{X} \in R^{L_1 \times L_2 \times \cdots \times L_M}$, $U \in R^{L'_d \times L_d}$, $V \in R^{L''_d \times L'_d}$, and VU is the standard matrix product of V and U.

Figure 3.6 shows a third order tensor $\mathbf{X} \in R^{L_1 \times L_2 \times L_3}$ mode–2 products with a matrix $U \in R^{L'_2 \times L_2}$. The result is a tensor in $R^{L_1 \times L'_2 \times L_3}$.

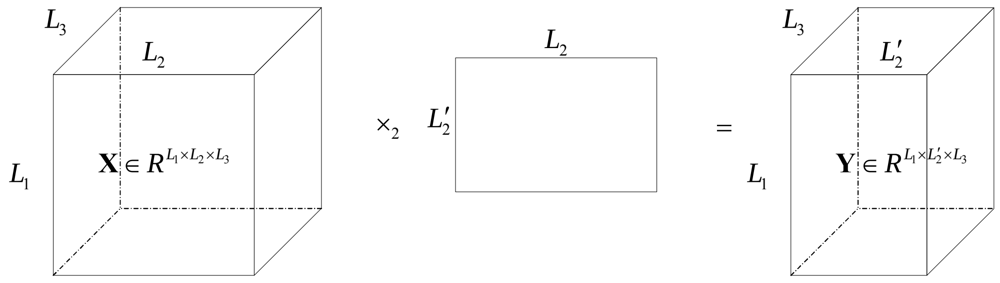

L_3
L_2
L_1
$\mathbf{X} \in R^{L_1 \times L_2 \times L_3}$
$\times_2$
L_2
L'_2
$=$
L_3
L'_2
L_1
$\mathbf{Y} \in R^{L_1 \times L'_2 \times L_3}$

Figure 3.6. The mode–2 product of a third order tensor $\mathbf{X} \in R^{L_1 \times L_2 \times L_3}$ and a matrix $U \in R^{L'_2 \times L_2}$ results in a new tensor $\mathbf{Y} \in R^{L_1 \times L'_2 \times L_3}$.

To simplify the notation in this thesis, we denote

$$\mathbf{X} \times_1 U_1 \times_2 U_2 \times \cdots \times_M U_M \triangleq \mathbf{X} \prod_{k=1}^{M} {}_{\times_k} U_k \tag{3.07}$$

and

$$\mathbf{X} \times_1 U_1 \times \cdots \times_{i-1} U_{i-1} \times_{i+1} U_{i+1} \times \cdots \times_M U_M = \mathbf{X} \prod_{d=1; d \neq i}^{M} {}_{\times_d} U_d \triangleq \mathbf{X} \bar{\times}_i U_i \,. \tag{3.08}$$

Definition 3.5 (Frobenius Norm for Tensor) The Frobenius norm of a tensor $\mathbf{X} \in R^{L_1 \times L_2 \times \cdots \times L_M}$ is given by

$$\|\mathbf{X}\|_{Fro} = \sqrt{\left[\mathbf{X} \otimes \mathbf{X}; (1:M)(1:M)\right]} = \sqrt{\sum_{l_1=1}^{L_1} \cdots \sum_{l_M=1}^{L_M} (\mathbf{X})^2_{l_1 \times \cdots \times l_M}} \,. \tag{3.09}$$

The Frobenius norm of a tensor $\mathbf{X}$ measures the "size" of the tensor and its square is the "energy" of the tensor.

Definition 3.6 (Rank–1 tensor) An M–th order tensor $\mathbf{X}$ has rank one if it is the tensor product of M vectors $\vec{u}_i \in R^{L_i}$, where $1 \le i \le M$:

$$\mathbf{X} = \vec{u}_1 \otimes \vec{u}_2 \otimes \cdots \otimes \vec{u}_M = \prod_{k=1}^{M} {}_{\otimes} \vec{u}_k . \tag{3.10}$$

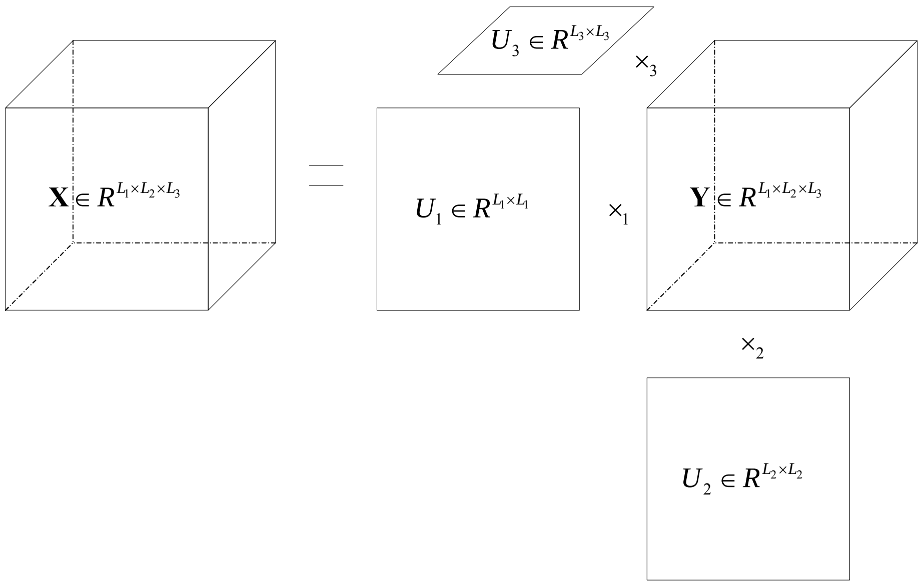
The rank of an arbitrary *M*–th order tensor $\mathbf{X}$, denoted by $R = \operatorname{rank}(\mathbf{X})$, is the minimum number of rank–1 tensors that yield $\mathbf{X}$ in a linear combination.

Definition 3.7 (*d*–rank) The *d*–rank of a tensor $\mathbf{X}$, represented by $R_d = \operatorname{rank}_d(\mathbf{X})$, is the dimension of the vector space generated by the mode–*d* matricizing.

Based on the definition of the *d*–rank of a tensor $\mathbf{X}$, we have

$$R_d = \operatorname{rank}_d(\mathbf{X}) = \operatorname{rank}(\operatorname{mat}_d(\mathbf{X})) = \operatorname{rank}(X_d). \tag{3.11}$$

The higher–order singular value decomposition (HOSVD) of a tensor is very important in tensor algebra. Most of the algorithms relevant to tensor algebra are based on HOSVD. HOSVD is a higher–order extension of the singular value decomposition (SVD) of a matrix. The HOSVD of a third order tensor $\mathbf{X} \in R^{L_1 \times L_2 \times L_3}$ is shown in Figure 3.7.



$\mathbf{X} \in R^{L_1 \times L_2 \times L_3}$
=
$U_3 \in R^{L_3 \times L_3}$
$\times_3$
$U_1 \in R^{L_1 \times L_1}$
$\times_1$
$\mathbf{Y} \in R^{L_1 \times L_2 \times L_3}$
$\times_2$
$U_2 \in R^{L_2 \times L_2}$

Figure 3.7. The HOSVD of a third order tensor $\mathbf{X} \in R^{L_1 \times L_2 \times L_3}$.

Theorem 3.1 (Higher–Order Singular Value Decomposition) [76]

A tensor $\mathbf{X} \in R^{L_1 \times L_2 \times \cdots \times L_M}$ can be decomposed as the product of a tensor $\mathbf{Y} \in R^{L_1 \times L_2 \times \cdots \times L_M}$ with a series of orthogonal matrices $U_k \in R^{L_k \times L_k}$, i.e.,

$$\mathbf{X} = \mathbf{Y} \prod_{k=1}^{M} \times_k U_k \ , \tag{3.12}$$

such that, the subtensor of $\mathbf{Y}_{l_k=\alpha} \in R^{L_1 \times \cdots L_{k-1} \times L_{k+1} \times L_M}$, obtained by fixing the kth ($1 \le l_k \le L_k$) index to α, is orthogonal to $\mathbf{Y}_{l_k=\beta} \in R^{L_1 \times \cdots L_{k-1} \times L_{k+1} \times L_M}$, i.e.,

$$\llbracket \mathbf{Y}_{l_k=\alpha} \otimes \mathbf{Y}_{l_k=\beta} ; (1:M-1)(1:M-1) \rrbracket = 0 . \tag{3.13}$$

when $\alpha \neq \beta$.

Finally,

$$\left\| \mathbf{Y}_{l_k=1} \right\|_{Fro} \ge \left\| \mathbf{Y}_{l_k=2} \right\|_{Fro} \ge \cdots \ge \left\| \mathbf{Y}_{l_k=L_k} \right\|_{Fro} \ge 0 . \tag{3.14}$$

Proof: See Appendix.

Table 3.1.
Alternating Projection for the Higher–Order Singular Value Decomposition.

Input: The input tensor $\mathbf{X} \in R^{L_1 \times L_2 \times \cdots \times L_M}$.

Output: Projection matrices $\left\{ U_k \mid_{k=1}^{M}, U_k \in R^{L_k \times L_k} \right\}$ and the tensor $\mathbf{Y} \in R^{L_1 \times L_2 \times \cdots \times L_M}$.

1.	Initialization: Set $U_k \mid_{k=1}^{M}$ be equal to random orthogonal matrices.
2.	*Conduct the following steps iteratively until convergence.*
3.	For $k=1$ to M
4.	Calculate $\mathbf{T} = \mathbf{X} \bar{\times}_k U_k^T$;
5.	Mode–*d* matricize $\mathbf{T}$ as $T_{(k)} \in R^{L_k \times \bar{L}_k}$ ($\bar{L}_k = L_1 \times \cdots L_{k-1} \times L_{k+1} \times L_M$);
6.	Calculate $S = T_{(k)} T_{(k)}^T$, $S \in R_{+}^{l_k \times l_k}$.
7.	Update the mode–*k* projection matrix U_k by conducting SVD on S: $S = U_k \Sigma_k U_k^T$.
8.	End
9.	Convergence checking: if $\left\| \left| U_{k(t-1)}^T U_{k(t)} \right| - I \right\|_{Fro} \le \varepsilon$ ($\varepsilon = 10^{-6}$) for all modes, the calculation has converged. Here $U_{k(t)}$ is the current optimized projection matrix and $U_{k(t-1)}$ is the previous projection matrix.
10.	Calculate the output tensor $\mathbf{Y} = \mathbf{X} \prod_{k=1}^{M} \times_k U_k^T$.

There is no closed form solution for HOSVD. In this thesis, an alternating projection method is used to obtain a solution for HOSVD, as shown in Table 3.1.

The Relationship between LSM and MLSM

Suppose: 1) we have a dimension reduction algorithm A_1, which finds a sequence of linear transformation matrices $U_i \in R^{L_i \times L_i'}$ ($L_i' < L_i$, $1 \le i \le M$) to transform a large tensor $\mathbf{X} \in R^{L_1 \times L_2 \times \cdots L_M}$ to a smaller tensor $\mathbf{Y}_1 \in R^{L_1' \times L_2' \times \cdots L_M'}$, i.e., $\mathbf{Y}_1 = \mathbf{X} \times_1 U_1^T \times_2 U_2^T \times \cdots \times_M U_M^T$; and 2) we have another dimension reduction algorithm A_2, which finds a linear transformation matrix $U \in R^{L \times L'}$ ($L = L_1 \times L_2 \times \cdots L_M$ and $L' = L_1' \times L_2' \times \cdots L_M'$; $L_i' < L_i$) to transform a high dimensional vector $\vec{x} = \operatorname{vect}(\mathbf{X})$ to a lower dimensional vector $\vec{y}_2 = \operatorname{vect}(\mathbf{Y}_2)$, i.e., $\vec{y}_2 = U^T \vec{x}$, where $\operatorname{vect}(\cdot)$ is the vectorization operator; $\vec{x} \in R^L$ and $\vec{y}_2 \in R^{L'}$. According to [181], we know

$$\begin{aligned} \vec{y}_1 &= \operatorname{vect}(\mathbf{Y}_1) \\ &= \operatorname{vect}\left(\mathbf{X} \times_1 U_1^T \times_2 U_2^T \times \cdots \times_M U_M^T\right) \\ &= \left(U_1 \otimes U_2 \otimes \cdots \otimes U_M\right)^T \operatorname{vect}(\mathbf{X}). \end{aligned} \tag{3.15}$$

Therefore, if $U = U_1 \otimes U_2 \otimes \cdots \otimes U_M$, $\vec{y}_2 = \vec{y}_1$. That is the algorithm A_1 is equal to the algorithm A_2, if the linear transformation matrix $U \in R^{L \times L'}$ in A_2 is equal to $U = U_1 \otimes U_2 \otimes \cdots \otimes U_M$ [11].

The tensor representation helps to reduce the number of parameters needed to model data. In A_1, there are $N_1 = \sum_{i=1}^{M} L_i L_i'$ parameters. While in A_2, there are $N_2 = \prod_{i=1}^{M} L_i \prod_{i=1}^{M} L_i'$ parameters. In statistical learning, we usually require that the number of the training samples is larger than the number of parameters to model these training samples for linear algorithms. In the training stage of the MLSM based learning algorithms, we usually use the alternating projection method to obtain a solution, i.e., the linear projection matrices are obtained independently, so we only need about $N_0 = \max_i \{L_i L_i'\}$ training samples to obtain a solution. However, N_2 training samples are required to obtain a

[11] In (3.15), we conduct the reshape operation on $\mathbf{U} = U_1 \otimes U_2 \otimes \cdots \otimes U_M$. That is, originally $\mathbf{U}$ lies in $R^{L_1 \times L_1' \times L_2 \times L_2' \times \cdots \times L_M \times L_M'}$ and after the reshape operation $\mathbf{U}$ is transformed to V in $R^{(L_1 \times L_2 \times \cdots \times L_M) \times (L_1' \times L_2' \times \cdots \times L_M')}$. Then, we can apply the transpose operation on V.

solution for LSM based learning algorithms. That is the MLSM based learning algorithms require a much smaller number of training samples than LSM based learning algorithms, because $N_0 \ll N_2$. Therefore, the tensor representation helps to reduce the small sample size (SSS) problem described in Chapter 2.

There is a long history of methods to reduce the number of parameters needed to model the data by introducing constraints. Take the methods in Gaussian distribution estimation as an example[12]: when the data consist of only a few training samples embedded in a high dimensional space, we always add some constraints to the covariance matrix, for example by requiring the covariance matrix to be a diagonal matrix. Therefore, to better characterize or classify natural data, a scheme should preserve as many as possible of the original constraints. When the training samples are limited, these constraints help to give reasonable solutions to classification problems.

Based on the discussions above, we have the following results:

1) when the number of the training samples is limited, the vectorization operation always leads to the SSS problem. That is, for a small size training set, we need to use the MLSM based learning algorithms, because the LSM based learning algorithms will over – fit the data. The vectorization of a tensor into a vector makes it harder to keep track of the information in spatial constraints. For example, two 4–neighbor connected pixels in an image may be widely separated from each other after a vectorization;

2) when the number of training samples is large, the MLSM based learning algorithms will under–fit the data. In this case, the vectorization operation for the data is helpful because it increases the number of parameters available to model the data.

[12] Constraints in MLSM are justified by the form of the data. However, constrains in the example are ad hoc.

Tensor Rank One Analysis

To study the tensor rank one analysis (TR1A) [132][13], we first introduce the best rank one approximation to a general tensor. That is, given an *M*–th order tensor $\mathbf{X} \in R^{L_1 \times L_2 \times \cdots L_M}$, find a scalar λ and a series of unit vectors $\vec{u}_k |_{k=1}^{M}$ to minimize the function f defined by

$$f\left(\vec{u}_k |_{k=1}^{M}\right) = \left\| \mathbf{X} - \lambda \prod_{k=1}^{M} {}_{\otimes} \vec{u}_k \right\|_{Fro}^{2}, \tag{3.16}$$

where $\lambda = \mathbf{X} \prod_{k=1}^{M} {}_{\times_k} \vec{u}_k^T$.

The Figure 3.8 shows the best rank one approximation of a third order tensor $\mathbf{X} \in R^{L_1 \times L_2 \times L_3}$.

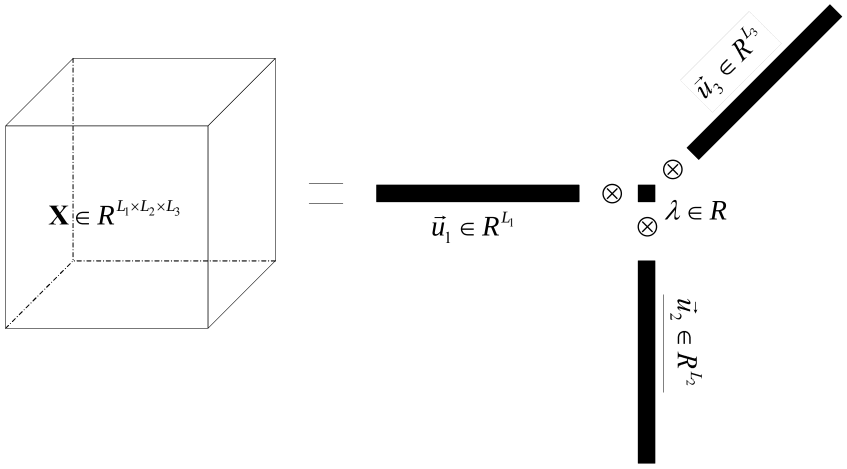


Figure 3.8. The best rank one approximation to a third order tensor $\mathbf{X} \in R^{L_1 \times L_2 \times L_3}$.

Theorem 3.2 [77] Minimizing $f\left(\vec{u}_k |_{k=1}^{M}\right) = \left\| \mathbf{X} - \lambda \prod_{k=1}^{M} {}_{\otimes} \vec{u}_k \right\|_{Fro}^{2}$ ($\lambda = \mathbf{X} \prod_{k=1}^{M} {}_{\times_k} \vec{u}_k^T$) is equivalent to maximizing

$$g\left(\vec{u}_k |_{k=1}^{M}\right) = \left\| \mathbf{X} \prod_{k=1}^{M} {}_{\times_k} \vec{u}_k^T \right\|_{Fro}^{2}. \tag{3.17}$$

Moreover, $f = \|\mathbf{X}\|_{Fro}^{2} - g$.

Proof: See Appendix.

[13] TR1A is an extension of the original algorithm, which only accepts matrices as input.

Based on (3.17), a solution of (3.16) can be obtained by the alternating projection method, listed in Table 3.2.

Table 3.2. Alternating Projection for the Best Rank One Approximation

Input: The input tensor $\mathbf{X} \in R^{L_1 \times L_2 \times \cdots L_M}$.

Output: Projection vectors $\left\{ \vec{u}_k |_{k=1}^{M}, \vec{u}_k \in R^{L_k} \right\}$ and the scalar λ, such that $f\left(\vec{u}_k |_{k=1}^{M}\right)$ is minimized.

1.	Initialization: Set $\vec{u}_k |_{k=1}^{M}$ be equal to random unit vectors.
2.	*Conduct the following steps iteratively until convergence.*
3.	For $k=1$ to M
4.	Calculate $\vec{v} = \mathbf{X} \bar{\times}_k \vec{u}_k^T$;
5.	Assignment: $\lambda \leftarrow \left\| \vec{v} \right\|$, $\vec{u}_k = \vec{v} / \lambda$.
6.	End
7.	Convergence checking: if $\left\| \left| \vec{u}_{k(t-1)}^T \vec{u}_{k(t)} \right| - 1 \right\| \le \varepsilon$ ($\varepsilon = 10^{-6}$) for all modes, the calculated $\vec{u}_k$ has converged. Here $\vec{u}_{k(t)}$ is the current optimized projection vector and $\vec{u}_{k(t-1)}$ is the previous projection vector.

TR1A is defined based on (3.16) for a number of samples $\mathbf{X}_i \in R^{L_1 \times L_2 \times \cdots L_M}$, $1 \le i \le N$ by minimizing the total reconstruction error through $\vec{u}_i |_{i=1}^{M}$ and $\lambda_i |_{i=1}^{N}$, i.e.,

$$f\left(\vec{u}_k |_{k=1}^{M}\right) = \sum_{i=1}^{N} \left\| \mathbf{X}_i - \lambda_i \prod_{k=1}^{M} {}_{\otimes} \vec{u}_k \right\|_{Fro}^2, \quad \lambda_i = \mathbf{X}_i \prod_{k=1}^{M} {}_{\times_k} \vec{u}_k^T. \tag{3.18}$$

Minimizing (3.18) is equivalent to maximizing

$$g\left(\vec{u}_k |_{k=1}^{M}\right) = \sum_{i=1}^{N} \left\| \mathbf{X}_i \prod_{k=1}^{M} {}_{\times_k} \vec{u}_k^T \right\|_{Fro}^2. \tag{3.19}$$

In practical applications, the total reconstruction error (3.18) is not small enough, so we need to use a number of rank one tensors $\vec{u}_{k,r} |_{1 \le k \le M}^{1 \le r \le R}$ and a number of scalars $\lambda_{i,r} |_{1 \le i \le N}^{1 \le r \le R}$ to approximate the original tensors $\mathbf{X}_i$ to minimize the function f defined by

$$f\left(\lambda_{i,r}\,|_{1\le i\le N}^{1\le r\le R},\vec{u}_{k,r}\,|_{1\le k\le M}^{1\le r\le R}\right)=\sum_{i=1}^{N}\left\|\mathbf{X}_i-\sum_{r=1}^{R}\lambda_{i,r}\prod_{k=1}^{M}\otimes\vec{u}_{k,r}\right\|_{Fro}^{2}. \qquad (3.20)$$

There is no closed form solution for (3.20). A solution, listed in Table 3.3, based on the alternating projection is suggested by Shashua and Levin [132].

(Note: The algorithm is based on the fact: minimizing $\sum_{i=1}^{N}\left\|\mathbf{X}_i-\lambda_i\prod_{k=1}^{M}\otimes\vec{u}_k\right\|_{Fro}^{2}$ is equivalent to maximizing $\sum_{i=1}^{N}\left\|\mathbf{X}_i\prod_{k=1}^{M}\times_k\vec{u}_k^T\right\|_{Fro}^{2}$.)

Table 3.3. Alternating Projection for the Tensor Rank One Analysis

Input: The input tensors $\mathbf{X}_i\in R^{L_1\times L_2\times\cdots L_M}$, $1\le i\le N$ and the number of rank one tensors R.

Output: Projection vectors $\left\{\vec{u}_{k,r}\,|_{1\le k\le M}^{1\le r\le R},\vec{u}_{k,r}\in R^{L_k}\right\}$ and scalars $\left\{\lambda_{i,r}\,|_{1\le i\le N}^{1\le r\le R},\lambda_{i,r}\in R\right\}$, such that (3.20) is minimized.

```
1.  For $r=1$ to $R$
2.      Set $\vec{u}_{k,r}|_{k=1}^{M}$ be equal to random unit vectors.
3.      Conduct steps 4–9 iteratively until convergence.
4.      For $k=1$ to $M$
5.          Calculate $\vec{v}_i=\mathbf{X}_i\bar{\times}_k\vec{u}_{k,r}^T$, $1\le i\le n$;
6.          Calculate the eigenvector $\vec{h}$ of $V_1=\sum_{i=1}^{n}\vec{v}_i\otimes\vec{v}_i$ associated with
            the largest eigenvalue; (This step is equivalent to calculating the
            left singular vector $\vec{h}$ of $V_2=[\vec{v}_1,\vec{v}_2,\cdots,\vec{v}_n]$ associated with the
            largest singular value.)
7.          Assignment: $\vec{u}_{k,r}\leftarrow\vec{h}$;
8.      End
9.      Convergence checking: if $\left|\left|\vec{u}_{k,r,t}^T\vec{u}_{k,r,t-1}\right|-1\right|\le\varepsilon$ ($\varepsilon=10^{-6}$) for all indices,
        the calculated $\vec{u}_{k,r}$ has converged. Here $\vec{u}_{k,r,t}$ is the current optimized
        projection vector and $\vec{u}_{k,r,t-1}$ is the previous projection vector.
10.     Assignment: $\lambda_{k,r}\leftarrow\mathbf{X}_i\prod_{k=1}^{M}\times_k\vec{u}_{k,r}^T$ ($1\le i\le N$);
11.     Assignment: $\mathbf{X}_i\leftarrow\mathbf{X}_i-\lambda_{k,r}\prod_{k=1}^{M}\otimes\vec{u}_{k,r}$ ($1\le i\le N$);
12. End
```

Based on the procedure to obtain a solution for TR1A listed in Table 3.3, we can calculate the scalars $\lambda_r |_{r=1}^{R}$ for any given tensor $\mathbf{X}$ through the following iterative procedure: 1). Calculate $\lambda_r = \mathbf{X} \prod_{k=1}^{M} \times_k \vec{u}_{k,r}^T$; 2) Assign $\mathbf{X} \leftarrow \mathbf{X} - \lambda_r \prod_{k=1}^{M} \otimes \vec{u}_{k,r}$, $1 \le r \le R$. Based on TR1A, a general tensor is represented by a sequence of rank one approximations.

General Tensor Analysis

General tensor analysis (GTA) [162] is a multidimensional extension of the principal component analysis (PCA). Before we describe GTA, the best $\text{rank}-(R_1, R_2, \cdots, R_M)$ approximation is introduced. The procedure is similar to the best rank one approximation.

Definition 3.8 [77] (Best Rank–$(R_1, R_2, \cdots R_M)$ Approximation) Given a tensor $\mathbf{X} \in R^{L_1 \times L_2 \times \cdots L_M}$, the tensor $\hat{\mathbf{X}} \in R^{L_1 \times L_2 \times \cdots L_M}$ with $\text{rank}\left(\text{mat}_d\left(\hat{\mathbf{X}}\right)\right) = R_d$, $1 \le d \le M$, which minimizes the least square cost $\left\|\mathbf{X} - \hat{\mathbf{X}}\right\|_{Fro}$, is the best rank–$(R_1, R_2, \cdots R_M)$ approximation of the original tensor $\mathbf{X}$.

The Figure 3.9 shows the best $\text{rank}-(R_1, R_2, R_3)$ approximation of a third order tensor $\mathbf{X} \in R^{L_1 \times L_2 \times L_3}$.

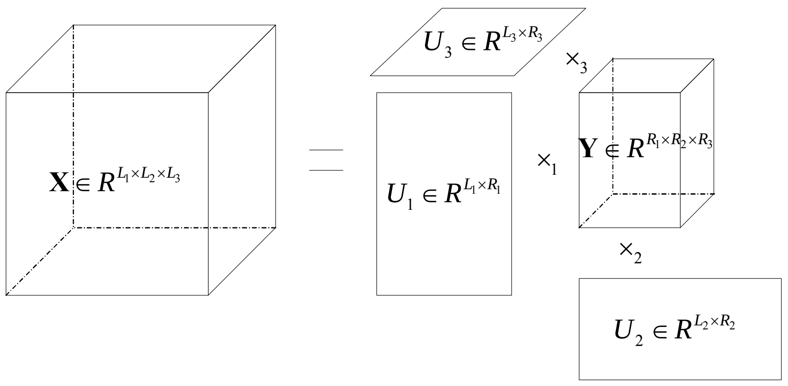

$\mathbf{X} \in R^{L_1 \times L_2 \times L_3}$
$=$
$U_3 \in R^{L_3 \times R_3}$
$\times_3$
$U_1 \in R^{L_1 \times R_1}$
$\times_1$
$\mathbf{Y} \in R^{R_1 \times R_2 \times R_3}$
$\times_2$
$U_2 \in R^{L_2 \times R_2}$

Figure 3.9. The best $\text{rank}-(R_1, R_2, R_3)$ approximation of a third order tensor $\mathbf{X} \in R^{L_1 \times L_2 \times L_3}$.

Theorem 3.3 [75] Given a sequence of unitary matrices $U_k \in R^{L_k \times L'_k}$ ($1 \le k \le M$ and $L'_d < L_d$) and a tensor $\mathbf{X} \in R^{L_1 \times L_2 \times \cdots L_M}$, the function $f\left(\hat{\mathbf{X}}\right) = \left\|\mathbf{X} - \hat{\mathbf{X}}\right\|_{Fro}^2$ is minimized, where $\text{rank}_d\left(\hat{\mathbf{X}}\right) = L'_d$ and $\hat{\mathbf{X}} \in R^{L_1 \times L_2 \times \cdots L_M}$

$$\hat{\mathbf{X}} = \left(\mathbf{X} \prod_{k=1}^{M} {}_{\times_k} U_k^T\right) \prod_{k=1}^{M} {}_{\times_k} U_k = \mathbf{X} \prod_{k=1}^{M} {}_{\times_k} \left(U_k U_k^T\right). \tag{3.21}$$

Proof: See Appendix.

Theorem 3.4 [75] For a given tensor $\mathbf{X} \in R^{L_1 \times L_2 \times \cdots L_M}$, minimizing $f\left(U_k \mid_{k=1}^{M}\right) = \left\| \mathbf{X} - \mathbf{X} \prod_{k=1}^{M} \times_k \left(U_k U_k^T\right) \right\|_{Fro}^2$ is equivalent to maximizing

$$g\left(U_k \mid_{k=1}^{M}\right) = \left\| \mathbf{X} \prod_{k=1}^{M} \times_k U_k^T \right\|_{Fro}^2 . \tag{3.22}$$

Proof: See Appendix.

Table 3.4.
Alternating Projection for the Best Rank$-\left(R_1, R_2, \cdots R_M\right)$ Approximation.

Input: The input tensors $\mathbf{X} \in R^{L_1 \times L_2 \times \cdots L_M}$ and the *d*–ranks $\left\{L'_d \mid_{d=1}^{M}\right\}$.

Output: Projection matrices $\left\{U_k \mid_{k=1}^{M}, U_k \in R^{L_k \times L'_k}\right\}$ and the tensor $\mathbf{Y} \in R^{L'_1 \times L'_2 \times \cdots \times L'_M}$, such that (3.22) is maximized.

```
1.  Initialization: Set $U_k \mid_{k=1}^{M}$ be equal to random matrices and each column of
    $U_k \mid_{k=1}^{M}$ is a unit vector.
2.  Conduct the following steps iteratively until convergence.
3.      For $k = 1$ to $M$
4.          Calculate $\mathbf{T} = \mathbf{X} \overline{\times}_k U_k^T$;
5.          Mode–d matricize $\mathbf{T}$ as $T_{(k)} \in R^{L_k \times \bar{L}_k}$
            ($\bar{L}_k = L_1 \times \cdots L_{k-1} \times L_{k+1} \times L_M$);
6.          Calculate $S = T_{(k)} T_{(k)}^T$, $S \in R_+^{L_k \times L_k}$.
7.          Update the mode–k projection matrix $U_k$ by the first $L'_k$ columns
            of the left singular vectors of $S$ associated with the $L'_k$ largest
            singular values;
8.      End
9.  Convergence checking: if $\left\| \left| U_{L,t-1}^T U_{L,t} \right| - I \right\|_{Fro} \le \varepsilon$ ($\varepsilon = 10^{-6}$) for all modes,
    the calculated $U_k$ has converged. Here $U_{k,t}$ is the current optimized
    projection matrix and $U_{k,t-1}$ is the previous projection matrix.
10. Calculate the core tensor $\mathbf{Y} = \mathbf{X} \prod_{k=1}^{M} \times_k U_k^T$.
```

There is no closed form solution for the best rank–$(R_1, R_2, \cdots R_M)$ approximation. The alternating projection method could be used to estimate this approximation based on (3.21) and (3.22) as listed in Table 3.4.

Table 3.5. Alternating Projection for General Tensor Analysis.

Input: The input tensor $\mathbf{X}_i \in R^{L_1 \times L_2 \times \cdots L_M}$, $1 \le i \le N$, and the *d*–ranks $\left\{ L'_d \,|_{d=1}^{M} \right\}$.

Output: Projection unitary matrices $\left\{ U_k \,|_{k=1}^{M}, U_k \in R^{L_k \times L'_k} \right\}$ and the tensor $\mathbf{Y}_i \in R^{L'_1 \times L'_2 \times \cdots \times L'_M}$, such that (3.23) is minimized.

Initialization: Set $U_k \,|_{k=1}^{M}$ be equal to random matrices and each column of $U_k \,|_{k=1}^{M}$ is a unit vector.

```
1. | Conduct the following steps iteratively until convergence.
2. |   For $k = 1$ to $M$
3. |       Calculate $\mathbf{T}_i = \mathbf{X}_i \bar{\times}_k U_k^T$, $1 \le i \le N$;
4. |       Mode–d matricize $\mathbf{T}_i$ as $T_{i;(k)} \in R^{L_k \times \bar{L}_k}$, $1 \le i \le N$,
   |       ($\bar{L}_k = L_1 \times \cdots L_{k-1} \times L_{k+1} \times L_M$);
5. |       Calculate $S = \sum_{i=1}^{N} T_{i;(k)} T_{i;(k)}^T$, $S \in R_{+}^{L_k \times L_k}$.
6. |       Update the mode–k projection matrix $U_k$ by the first $L'_k$ columns
   |       of the left singular vectors of $S$ associated with the $L'_k$ largest
   |       singular values;
7. |   End
8. | Convergence checking: if $\left\| \left| U_{k,t-1}^T U_{k,t} \right| - I \right\|_{Fro} \le \varepsilon$ ($\varepsilon = 10^{-6}$) for all modes,
   | the calculated $U_k$ has converged. Here $U_{k,t}$ is the current optimized
   | projection matrix and $U_{k,t-1}$ is the previous projection matrix.
9. | Calculate the core tensor $\mathbf{Y}_i = \mathbf{X}_i \prod_{k=1}^{M} \times_k U_k^T$.
```

GTA is defined based on (3.21). Given a number of samples $\mathbf{X}_i \in R^{L_1 \times L_2 \times \cdots L_M}$, $1 \le i \le N$, GTA minimizes the function f defined by

$$f\left(U_k \,|_{k=1}^{M}\right) = \sum_{i=1}^{N} \left\| \mathbf{X}_i - \mathbf{Y}_i \prod_{k=1}^{M} \times_k U_k \right\|_{Fro}^2, \tag{3.23}$$

where $\mathbf{Y}_i = \mathbf{X}_i \prod_{k=1}^{M} \times_k U_k^T$.

Based on Theorem 3.4, minimizing (3.23) is equivalent to maximizing

$$g\left(U_k \mid_{k=1}^{M}\right) = \sum_{i=1}^{N} \left\| \mathbf{X}_i \prod_{k=1}^{M} \times_k U_k^T \right\|_{Fro}^2 . \tag{3.24}$$

There is no closed form solution to (3.23). A solution, listed in Table 3.5, based on the alternating projection is similar to Table 3.3.

(Note: The algorithm is based on the fact: minimizing $\sum_{i=1}^{N} \left\| \mathbf{X}_i - \mathbf{Y}_i \prod_{k=1}^{M} \times_k U_k \right\|_{Fro}^2$ is equivalent to maximizing $\sum_{i=1}^{N} \left\| \mathbf{X}_i \prod_{k=1}^{M} \times_k U_k^T \right\|_{Fro}^2$.)

Two Dimensional Linear Discriminant Analysis

The major difference between LDA and 2DLDA is the representation of the training samples. In LDA, the training samples are represented by vectors, while they are represented as matrices in 2DLDA.

Suppose: the training samples are divided into C classes and the ith ($1 \le i \le C$) class contains N_i samples $X_{i;j} \in R^{L_1 \times L_2}$ ($1 \le j \le N_i$). The mean matrix of all training samples is $M = \left(\sum_{i=1}^{C}\sum_{j=1}^{N_i} X_{i;j}\right) \Big/ \left(\sum_{i=1}^{C} N_i\right)$ and the mean matrix of the ith class samples is $M_i = \left(\sum_{j=1}^{N_i} X_{i;j}\right) \Big/ N_i$. The class structure in the high dimensional space is defined by

$$\begin{aligned} D_w^{(H)} &= \mathrm{tr}\left(\sum_{i=1}^{C}\sum_{j=1}^{N_i}\left(X_{i;j}-M_i\right)\left(X_{i;j}-M_i\right)^T\right) \\ D_b^{(H)} &= \mathrm{tr}\left(\sum_{i=1}^{C} N_i\left(M_i-M\right)\left(M_i-M\right)^T\right), \end{aligned} \tag{3.25}$$

The class structure in the low dimensional space is defined by

$$\begin{aligned} D_w^{(L)} &= \mathrm{tr}\left(\sum_{i=1}^{C}\sum_{j=1}^{N_i} U^T\left(X_{i;j}-M_i\right)VV^T\left(X_{i;j}-M_i\right)^T U\right) \\ D_b^{(L)} &= \mathrm{tr}\left(\sum_{i=1}^{C} N_i U^T\left(M_i-M\right)VV^T\left(M_i-M\right)^T U\right), \end{aligned} \tag{3.26}$$

The 2DLDA finds two projection matrices $U \in R^{L_1 \times L_1'}$ and $V \in R^{L_2 \times L_2'}$ to preserve the class structure of the original high dimensional space in the projected low dimensional space by maximizing $\left(D_w^{(L)}\right)^{-1} D_b^{(L)}$. Ye et al. [173] developed the following alternating projection method to obtain a solution of 2DLDA. The algorithm conducts the following two steps iteratively:

Step 1: calculate U, which is the L_1' eigenvectors of $A_w^{-1}A_b$ associated with the largest L_1' eigenvalues, with the given V (it is a random matrix and each column is a unit vector), where A_w and A_b are defined by

$$A_w = \sum_{i=1}^{C}\sum_{j=1}^{N_i}\left(X_{i;j} - M_i\right)VV^T\left(X_{i;j} - M_i\right)^T$$
$$A_b = \sum_{i=1}^{C} N_i\left(M_i - M\right)VV^T\left(M_i - M\right)^T. \tag{3.27}$$

Step 2: calculate V, which is the L_2' eigenvectors of $B_w^{-1}B_b$ associated with the largest L_2' eigenvalues, with the given U (obtained in Step 1), where B_w and B_b are defined by

$$B_w = \sum_{i=1}^{C}\sum_{j=1}^{N_i}\left(X_{i;j} - M_i\right)^T UU^T\left(X_{i;j} - M_i\right)$$
$$B_b = \sum_{i=1}^{C} N_i\left(M_i - M\right)^T UU^T\left(M_i - M\right). \tag{3.28}$$

The 2DLDA reduces the SSS problem and it is more efficient than LDA in terms of time complexity and space complexity. The drawback of 2DLDA is the alternating projection algorithm to obtain a solution of 2DLDA does not converge. Therefore, the recognition accuracies are not stable over different training iterations. For example, in the Probe A of the human gait recognition task in this Chapter, the rank one recognition rate changes from 75% to 91% in the first 100 training iterations. This is because the projection matrices U and V do not maximize $\left(D_w^{(L)}\right)^{-1} D_b^{(L)}$. According to steps 1 and 2, the projection matrix U is the L_1' eigenvectors of $A_w^{-1}A_b$ associated with the largest L_1' eigenvalues, which is obtained with the given V by maximizing $\mathrm{tr}\left(\left(U^T A_w U\right)^{-1} U^T A_b U\right)$ while the projection matrix V is the L_2' eigenvectors of $B_w^{-1}B_b$ associated with the largest L_2' eigenvalues, which is obtained with the given U by maximizing $\mathrm{tr}\left(\left(U^T B_w U\right)^{-1} U^T B_b U\right)$. The projection matrices U and V, obtained by iteratively conducting steps 1 and 2, do not maximize $\left(D_w^{(L)}\right)^{-1} D_b^{(L)}$, because maximizing the trace ratio between two projected matrices does not be equal to maximizing the ratio trace between two projected matrices [103], i.e., with the given A_w and A_b, the argument of maximum $\mathrm{tr}\left(\left(U^T A_w U\right)^{-1} U^T A_b U\right)$ by varying U does not maximize $\mathrm{tr}\left(U^T A_b U\right)/\mathrm{tr}\left(U^T A_w U\right)$ and with the given B_w

and B_b, the argument of maximum $\operatorname{tr}\left(\left(V^T B_w V\right)^{-1} V^T B_b V\right)$ by varying V does not maximize $\operatorname{tr}\left(V^T B_b V\right) / \operatorname{tr}\left(V^T B_w V\right)$.

To deal with the nonconvergence problem and to accept general tensors as input for discriminative dimension reduction, we propose GTDA in the next Section.

General Tensor Discriminant Analysis

Although the effectiveness and the efficiency of 2DLDA, as a preprocessing for subsequent classification, e.g., by LDA, have been proved, 2DLDA fails to converge in the training stage. The nonconvergence shadows the advantages of 2DLDA, because: 1) it is difficult to determine when to stop the training stage; and 2) different numbers of training iterations will lead to different recognition results. To solve the nonconvergence problem, we develop GTDA, which 1) provides a converged alternating projection algorithm for training; 2) accepts general tensors as input; 3) preserves the discriminative information for classification; and 4) reduces the SSS problem in the subsequent classification, e.g., by LDA. The proposed GTDA is based on the differential scatter discriminant criterion (DCDS).

Differential Scatter Discriminant Criterion (DSDC)

The Differential Scatter Discriminant Criterion (DSDC) [39][149][143][145] is defined by,

$$U^* = \arg\max_{U^T U = I} \left(\operatorname{tr}\left(U^T S_b U\right) - \zeta \operatorname{tr}\left(U^T S_w U\right) \right), \tag{3.29}$$

where ζ is a tuning parameter; $U \in R^{L \times L'}$ ($L' < L$) , constrained by $U^T U = I$, is the projection matrix; and S_b and S_w are defined in (2.10).

According to [39] (pp. 446 – 447), the solution to (3.29) is equivalent to the solution to (12) for some special ζ in (3.29). If we extract only one feature, i.e., U degenerates to a vector, then $\zeta = \lambda_{\max}\left(S_w^{-1} S_b\right)$, which is the maximum eigenvalue of $S_w^{-1} S_b$. If we want to extract L' features simultaneously, we estimate ζ as $\sum_{i=1}^{L'} \lambda_i$, where $\lambda_i |_{i=1}^{L'}$ are the largest L' eigenvalues of $S_w^{-1} S_b$. From [39] (pp. 446 – 447), it is not difficult to show that a suitable ζ in (3.29) is $\operatorname{tr}\left(U_{opt}^T S_b U_{opt}\right) \big/ \operatorname{tr}\left(U_{opt}^T S_w U_{opt}\right)$[14]. An accurate solution of (3.29) can be obtained by

[14] The derivative of $\operatorname{tr}\left(U^T S_b U\right) - \zeta \operatorname{tr}\left(U^T S_w U\right)$ with respect to U is given by $S_b U - \zeta S_w U$. To obtain a solution of (3.29), we need to set $S_b U - \zeta S_w U = 0$ (as we have a strict condition here,

an alternating projection method. Here, we use the approximation (on setting ζ as the maximum eigenvalue of $S_w^{-1} S_b$) in (3.29) to avoid the alternating projection method for optimization.

In realworld applications, because the distribution of testing samples diverges from the distribution of training samples, a manually chosen value of ζ always achieves better prediction results than the calculated value. However, the manual setting of ζ is not practical for real applications, because we do not know the best choice of ζ for classification. In this thesis, ζ is selected automatically during the training procedure.

General Tensor Discriminant Analysis

On defining S_b and S_w by (2.10), it follows from (3.29) that

$$
\begin{aligned}
U^* &= \arg\max_{U^T U = I} \left(\operatorname{tr}\left(U^T S_b U\right) - \zeta \operatorname{tr}\left(U^T S_w U\right) \right) \\
&= \arg\max_{U^T U = I} \begin{pmatrix} \sum_{i=1}^{C} N_i \llbracket \left((\vec{m}_i - \vec{m}) \times_1 U^T \right) \otimes \left((\vec{m}_i - \vec{m}) \times_1 U^T \right); (1)(1) \rrbracket \\ -\zeta \sum_{i=1}^{C} \sum_{j=1}^{N_i} \llbracket \left((\vec{x}_{i;j} - \vec{m}_i) \times_1 U^T \right) \otimes \left((\vec{x}_{i;j} - \vec{m}_i) \times_1 U^T \right); (1)(1) \rrbracket \end{pmatrix} \qquad (3.30) \\
&= \arg\max_{U^T U = I} \left(\sum_{i=1}^{C} N_i \left\| (\vec{m}_i - \vec{m}) \times_1 U^T \right\|_{Fro}^2 - \zeta \sum_{i=1}^{c} \sum_{j=1}^{n_i} \left\| (\vec{x}_{i;j} - \vec{m}_i) \times_1 U^T \right\|_{Fro}^2 \right)
\end{aligned}
$$

where $\|\cdot\|_{Fro}$ is the Frobenius norm and the projection matrix $U \in R^{L \times L'}$ ($L' < L$) is constrained by $U^T U = I$.

Deduction 3.1: See Appendix.

Let $\mathbf{X}_{i;j}$ denote the *j*th training sample (tensor) in the *i*th individual class, $\mathbf{M}_i = (1/N_i) \sum_{j=1}^{N_i} \mathbf{X}_{i;j}$ is the mean tensor of the samples in the *i*th class, $\mathbf{M} = (1/N) \sum_{i=1}^{C} N_i \mathbf{M}_i$ is the mean tensor of all training samples, and U_k denotes the *k*th direction projection matrix for decomposition in the training procedure. Moreover, $\mathbf{X}_{i;j} |_{1 \le i \le C}^{1 \le j \le N_i}$, $\mathbf{M}_i |_{i=1}^{C}$, and $\mathbf{M}$ are all *M*–th order tensors in

i.e., $(S_b - \zeta S_w) u_k = 0$, $\forall u_k \in U$, u_k is a column vector in U). Consequently, we have $\operatorname{Tr}\left(U_{opt}^T S_b U_{opt}\right) = \zeta \operatorname{Tr}\left(U_{opt}^T S_w U_{opt}\right)$, where U_{opt} is a solution to (3.29).

$R^{L_1 \times L_2 \times \cdots \times L_M}$. Based on (3.30), we analogously define GTDA by replacing $\vec{x}_{i;j}$, $\vec{m}_i$, and $\vec{m}$ with $\mathbf{X}_{i;j}$, $\mathbf{M}_i$, and $\mathbf{M}$, respectively, as:

$$U_l^* |_{l=1}^M = \underset{U_k^T U_k = I|_{k=1}^M}{\arg\max} \left(\begin{array}{l} \sum_{i=1}^{C} N_i \left[\begin{array}{l} \left((\mathbf{M}_i - \mathbf{M}) \prod_{k=1}^{M} \times_k U_k^T \right) \\ \otimes \left((\mathbf{M}_i - \mathbf{M}) \prod_{k=1}^{M} \times_k U_k^T \right) \end{array} ; (1:M)(1:M) \right] \\ -\zeta \sum_{i=1}^{C} \sum_{j=1}^{N_i} \left[\begin{array}{l} \left((\mathbf{X}_{i;j} - \mathbf{M}_i) \prod_{k=1}^{M} \times_k U_k^T \right) \\ \otimes \left((\mathbf{X}_{i;j} - \mathbf{M}_i) \prod_{k=1}^{M} \times_k U_k^T \right) \end{array} ; (1:M)(1:M) \right] \end{array} \right). \tag{3.31}$$

According to the definition of the Frobenius norm of a tensor in (3.09), the right hand side of (3.31) can be simplified as

$$\begin{aligned} & U_l^* |_{l=1}^M \\ & = \underset{U_k^T U_k = I|_{k=1}^M}{\arg\max} \left(\sum_{i=1}^{C} N_i \left\| (\mathbf{M}_i - \mathbf{M}) \prod_{k=1}^{M} \times_k U_k^T \right\|_{Fro}^2 - \zeta \sum_{i=1}^{C} \sum_{j=1}^{N_i} \left\| (\mathbf{X}_{i;j} - \mathbf{M}_i) \prod_{k=1}^{M} \times_k U_k^T \right\|_{Fro}^2 \right). \end{aligned} \tag{3.32}$$

There is no closed form solution to (3.31), so we choose to use an alternating projection method, which is an iterative procedure, to obtain a numerical solution. Therefore, (3.31) is decomposed into *M* different optimization sub–problems, as follows,

$$\begin{aligned} U_l^* &= \underset{U_l^T U_l = I}{\arg\max} \left(\begin{array}{l} \sum_{i=1}^{C} N_i \left[\begin{array}{l} \left((\mathbf{M}_i - \mathbf{M}) \prod_{k=1}^{M} \times_k U_k^T \right) \\ \otimes \left((\mathbf{M}_i - \mathbf{M}) \prod_{k=1}^{M} \times_k U_k^T \right) \end{array} ; (1:M)(1:M) \right] \\ -\zeta \sum_{i=1}^{C} \sum_{j=1}^{N_i} \left[\begin{array}{l} \left((\mathbf{X}_{i;j} - \mathbf{M}_i) \prod_{k=1}^{M} \times_k U_k^T \right) \\ \otimes \left((\mathbf{X}_{i;j} - \mathbf{M}_i) \prod_{k=1}^{M} \times_k U_k^T \right) \end{array} ; (1:M)(1:M) \right] \end{array} \right) \\ &= \underset{U_l^T U_l = I}{\arg\max} \, \mathrm{tr} \left(U_l^T \left(\begin{array}{l} \sum_{i=1}^{C} \left[\begin{array}{l} n_i \, \mathrm{mat}_l \left((\mathbf{M}_i - \mathbf{M}) \bar{\times}_l U_l^T \right) \\ \mathrm{mat}_l^T \left((\mathbf{M}_i - \mathbf{M}) \bar{\times}_l U_l^T \right) \end{array} \right] \\ -\zeta \sum_{i=1}^{C} \sum_{j=1}^{N_i} \left[\begin{array}{l} \mathrm{mat}_l \left((\mathbf{X}_{i;j} - \mathbf{M}_i) \bar{\times}_l U_l^T \right) \\ \mathrm{mat}_l^T \left((\mathbf{X}_{i;j} - \mathbf{M}_i) \bar{\times}_l U_l^T \right) \end{array} \right] \end{array} \right) U_l \right). \end{aligned} \tag{3.33}$$

Deduction 3.2: See Appendix.

To simplify (3.33), we define

$$B_l = \sum_{i=1}^{C} \left[N_i \operatorname{mat}_l \left(\left(\mathbf{M}_i - \mathbf{M} \right) \bar{\times}_l U_l^T \right) \operatorname{mat}_l^T \left(\left(\mathbf{M}_i - \mathbf{M} \right) \bar{\times}_l U_l^T \right) \right], \tag{3.34}$$

and

$$W_l = \sum_{i=1}^{C} \sum_{j=1}^{N_i} \left[\operatorname{mat}_l \left(\left(\mathbf{X}_{i;j} - \mathbf{M}_i \right) \bar{\times}_l U_l^T \right) \operatorname{mat}_l^T \left(\left(\mathbf{X}_{i;j} - \mathbf{M}_i \right) \bar{\times}_l U_l^T \right) \right]. \tag{3.35}$$

Therefore, (3.33) is simplified as,

$$U_l^* |_{l=1}^{M} = \underset{U_l^T U_l = I|_{l=1}^{M}}{\arg\max} \operatorname{tr}\left(U_l^T \left(B_l - \zeta W_l \right) U_l \right). \tag{3.36}$$

As pointed out in 0, ζ is the tuning parameter.

Table 3.6 lists the alternating projection method based optimization procedure for GTDA with the given tuning parameter ζ to simplify the proof of the convergence theorem. Later, we describe how to determine the tuning parameter ζ and the dimension $L_1' \times L_2' \times \cdots L_M'$ of the output tensors automatically. The key steps in the alternating projection procedure are Steps 3 – 5, which involve finding the lth mode projection matrix $U_{l,t}$ in the tth iteration using $U_{k,t} |_{k=1}^{l-1}$ obtained in the tth iteration and $U_{k,t-1} |_{k=l+1}^{M}$ obtained in the $(t-1)$th iteration. In Steps 3 and 4, we obtain the between class scatter matrix $B_{l,t-1}$ and the within class scatter matrix $W_{l,t-1}$ with $U_{k,t} |_{k=1}^{l-1}$ obtained in the tth iteration and $U_{k,t-1} |_{k=l+1}^{M}$ obtained in the $(t-1)$th iteration. The singular value decomposition (SVD) of $B_{l,t-1} - \zeta W_{l,t-1}$ is obtained and $U_{l,t-1}$ is updated using the eigenvectors of $B_{l,t-1} - \zeta W_{l,t-1}$, which correspond to the largest eigenvalues of $B_{l,t-1} - \zeta W_{l,t-1}$.

According to the algorithm described in Table 3.6, we can obtain a solution $U_k |_{k=1}^{M} \in R^{L_k \times L_k'}$ ($L_k' < L_k$) by an iterative way. For GTDA, we use the projected tensor $\mathbf{Y} = \mathbf{X} \prod_{k=1}^{M} \times_k U_k^T$ to represent the original general tensor $\mathbf{X}$. Unlike 2DLDA [173], the alternating projection method based optimization procedure for GTDA converges, as proved in Theorem 3.5.

Theorem 3.5 The alternating projection method based optimization procedure for GTDA converges.

Proof: See Appendix.

Table 3.6. Alternating Projection for General Tensor Discriminant Analysis

Input: Training tensors $\mathbf{X}_{i;j}\,|_{1\le i\le C}^{1\le j\le N_i} \in R^{L_1\times L_2\times\cdots\times L_M}$, the dimension of the output tensors $\mathbf{Y}_{i;j} \in R^{L_1'\times L_2'\times\cdots\times L_M'}$, the tuning parameters ζ, and the maximum number of training iterations T.

Output: The projection matrix $U_k\,|_{k=1}^{M} \in R^{L_k'\times L_k}$ ($U_k^T U_k = I$) and the output tensors $\mathbf{Y}_{i;j}\,|_{1\le i\le C}^{1\le j\le N_i} \in R^{L_1'\times L_2'\times\cdots\times L_M'}$.

```
1.  Initialize: Set $U_{k,0}\,|_{k=1}^{M} \in R^{L_k\times L_k'}$ be equal to random matrices and each
    column of $U_{k,0}\,|_{k=1}^{M}$ is a unit vector.
2.  For $t=1$ to $T$ {
3.      For $l=1$ to $M$ {
4.          Calculate $B_{l,t-1} = \sum_{i=1}^{C}\left[N_i \operatorname{mat}_l\left(\left(\mathbf{M}_i-\mathbf{M}\right)\overline{\times}_l U_{l,t-1}^T\right) \operatorname{mat}_l^T\left(\left(\mathbf{M}_i-\mathbf{M}\right)\overline{\times}_l U_{l,t-1}^T\right)\right]$
5.          Calculate $W_{l,t-1} = \sum_{i=1}^{C}\sum_{j=1}^{N_i}\left[\operatorname{mat}_l\left(\left(\mathbf{X}_{i;j}-\mathbf{M}_i\right)\overline{\times}_l U_{l,t-1}^T\right) \operatorname{mat}_l^T\left(\left(\mathbf{X}_{i;j}-\mathbf{M}_i\right)\overline{\times}_l U_{l,t-1}^T\right)\right]$
6.          Optimize $U_{l,t} = \arg\max_{U^T U=I} \operatorname{tr}\left(U^T\left(B_{l,t-1}-\zeta W_{l,t-1}\right)U\right)$ by SVD on
            $B_{l,t-1}-\zeta W_{l,t-1}$.
7.      }//For loop in Step 2.
8.      Convergence check: the training stage of GTDA converges if
        $\operatorname{Err}(t)=\sum_{l=1}^{M}\left\|\left|U_{l,t}^T U_{l,t-1}\right|-I\right\| \le \varepsilon$ ($\varepsilon = 10^{-6}$).
9.  }// For loop in Step 1.
10. $\mathbf{Y}_{i;j} = \mathbf{X}_{i;j}\prod_{k=1}^{M}\times_k U_k^T$.
```

For practical applications, it is important to determine the tuning parameter ζ and the dimension $L_1'\times L_2'\times\cdots\times L_M'$ of the output tensors automatically. In the tth training iteration and the lth order, we adjust ζ after Step 4 and before Step 5 by setting ζ_t being equal to the maximum eigenvalue of $W_{l,t-1}^{-1}B_{l,t-1}$. That is ζ is varying from one iteration to the next. In the tth training iteration, the lth dimension of the output tensors L_l' is determined by the lth projection matrix $U_{l,t}$, so we set a threshold value δ to automatically determine L_l' according to the following inequality:

$$\begin{aligned}
&\frac{\lambda_{l,1,t}}{\sum_{j=1}^{L_l} \lambda_{l,j,t}} < \frac{\lambda_{l,1,t} + \lambda_{l,2,t}}{\sum_{j=1}^{L_l} \lambda_{l,j,t}} < \cdots \\
&\quad < \frac{\sum_{i=1}^{L_l'} \lambda_{l,i,t}}{\sum_{j=1}^{L_l} \lambda_{l,j,t}} \le \delta < \frac{\sum_{i=1}^{L'+1_l} \lambda_{l,i,t}}{\sum_{j=1}^{L_l} \lambda_{l,j,t}} < \cdots . \\
&\qquad < \frac{\sum_{i=1}^{L_l} \lambda_{l,i,t}}{\sum_{j=1}^{L_l} \lambda_{l,j,t}} = 1,
\end{aligned} \tag{3.37}$$

where $\lambda_{l,i,t}$ is the i^{th} eigenvalue of $B_{l,t-1} - \zeta_t W_{l,t-1}$ and $\lambda_{l,i,t} \ge \lambda_{l,j,t}$ if $i < j$. Therefore, there is only one parameter, the threshold value δ, which needs to be tuned for recognition tasks. Without this method as shown in (3.37), we have to tune $M+1$ parameters, comprising of one parameter for each order of the *M*–th order tensors and ζ in (3.36). This multiparameter tuning is too time consuming when *M* is large.

Computational Complexity

The time complexity of LDA is $O\left(\left(\prod_{i=1}^{M} L_i\right)^3\right)$ in the training stage, where training samples $\mathbf{X}$ belong to $R^{L_1 \times L_2 \times \cdots \times L_M}$. The time complexity of the alternating projection method based optimization procedure of GTDA is $O\left(T \sum_{i=1}^{M} L_i^3\right)$, where T is the number of iterations required for GTDA to converge. The space complexity of LDA is $O\left(\left(\prod_{i=1}^{M} L_i\right)^2\right)$ in the training stage. The space complexity of the alternating projection method based optimization procedure of GTDA is $O\left(\sum_{i=1}^{M} L_i^2\right)$.

Manifold Learning using Tensor Representations

The proposed GTDA can also be extended for manifold learning [13], which has become popular in machine learning. Manifold learning is based on geometrical assumptions, i.e., a data set approximately lies in a low dimensional manifold embedded in the original high dimensional feature space. Recently, a large number of algorithms have been developed based on different criteria. For example, ISOMAP [155] finds the low dimensional representation for data by preserving the geodesic distances of data pairs in the original high dimensional space. Locally linear embedding (LLE) [124][127] produces the low dimensional representation for locally sufficient reconstruction, i.e., a sample is reconstructed from its neighborhood samples. Laplacian Eigenmap (LE) [5][6]reduces the data dimension by preserving the locality character of the samples. Recently, Bengio et al. [8] unified a number of manifold learning algorithms together and developed the out of sample extension. Yan et al. [172] developed another framework, named the graph embedding framework (GEF). GEF finds the projection matrix $U \in R^{L \times L'}$ ($L' < L$) to map the original high dimensional samples $\vec{x} \in R^{L}$ to the low dimensional space $R^{L'}$ through a linear projection, i.e., $\vec{y} = U^T \vec{x}$ and $\vec{y} \in R^{L'}$. The objective function of GEF is defined by

$$F(U) = \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} b_{ij} \left\| U^T \left(\vec{x}_i - \vec{x}_j \right) \right\|_{Fro}^2}{\sum_{i=1}^{N} \sum_{j=1}^{N} s_{ij} \left\| U^T \left(\vec{x}_i - \vec{x}_j \right) \right\|_{Fro}^2}, \tag{3.38}$$

where s_{ij} and b_{ij} are predefined weighting factors and N is the number of training samples. By varying s_{ij} and b_{ij}, different dimension reduction algorithms can be obtained, e.g., LLE, ISOMAP, and LDA. The linear projection matrix $U \in R^{L \times L'}$ is obtained by maximizing $F(U)$ subject to the constraint $U^T U = I$. To extend GEF to accept general tensors as input, we reformulate (3.38) as

$$
\begin{aligned}
G(U) &= \sum_{i=1}^{n}\sum_{j=1}^{n} b_{ij}\left\| U^T\left(\vec{x}_i-\vec{x}_j\right)\right\|_{Fro}^2 - \zeta\sum_{i=1}^{n}\sum_{j=1}^{n} s_{ij}\left\| U^T\left(\vec{x}_i-\vec{x}_j\right)\right\|_{Fro}^2 \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}\left(b_{ij}-\zeta s_{ij}\right)\left\| U^T\left(\vec{x}_i-\vec{x}_j\right)\right\|_{Fro}^2 \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}\left(b_{ij}-\zeta s_{ij}\right)\left\| \left(\vec{x}_i-\vec{x}_j\right)\times_1 U^T\right\|_{Fro}^2 \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}\left(b_{ij}-\zeta s_{ij}\right)\left\llbracket \begin{array}{l}\left(\left(\vec{x}_i-\vec{x}_j\right)\times_1 U^T\right)\\ \otimes\left(\left(\vec{x}_i-\vec{x}_j\right)\times_1 U^T\right)\end{array} ;(1)(1)\right\rrbracket,
\end{aligned}
\tag{3.39}
$$

where ζ is a tuning parameter and the linear projection matrix U is obtained by maximizing $G(U)$ constrained by $U^T U = I$.

Based on (3.33), we analogously define tensorized GEF by replacing $\vec{x}_i$ with $\mathbf{X}_i$ as

$$
H\left(U_k\,|_{k=1}^{M}\right) = \sum_{i=1}^{N}\sum_{j=1}^{N}\left(b_{ij}-\zeta s_{ij}\right)\left\llbracket \begin{array}{l}\left(\left(\mathbf{X}_i-\mathbf{X}_j\right)\prod_{k=1}^{M}\times_k U_k^T\right)\\ \otimes\left(\left(\mathbf{X}_i-\mathbf{X}_j\right)\prod_{k=1}^{M}\times_k U_k^T\right)\end{array} ;(1:M)(1:M)\right\rrbracket, \tag{3.40}
$$

where ζ is a tuning parameter and a series of linear projection matrices $U_k\,|_{k=1}^{M}$ is obtained by maximizing $H\left(U_k\,|_{k=1}^{M}\right)$ constrained by $U_k^T U_k = I$ for $1 \le k \le M$. According to (3.09), the tensorized GEF can be defined as

$$
\begin{aligned}
U_k^*\,|_{k=1}^{M} &= \underset{U_k^T U_k = I|_{k=1}^{M}}{\arg\max}\, H\left(U_k\,|_{k=1}^{M}\right) \\
&= \underset{U_k^T U_k = I|_{k=1}^{M}}{\arg\max}\left(\sum_{i=1}^{N}\sum_{j=1}^{N}\left(b_{ij}-\zeta s_{ij}\right)\left\|\left(\mathbf{X}_i-\mathbf{X}_j\right)\prod_{k=1}^{M}\times_k U_k^T\right\|_{Fro}^2\right).
\end{aligned}
\tag{3.41}
$$

Similar to GTDA, there is no closed form solution. Therefore, (3.41) is decomposed into *M* different optimization sub–problems, as follows,

$$
\begin{aligned}
U_k^* &= \underset{U_k^T U_k = I}{\arg\max}\left(\sum_{i=1}^{N}\sum_{j=1}^{N}\left(b_{ij}-\zeta s_{ij}\right)\left\llbracket \begin{array}{l}\left(\left(\mathbf{X}_i-\mathbf{X}_j\right)\prod_{k=1}^{M}\times_k U_k^T\right)\\ \otimes\left(\left(\mathbf{X}_i-\mathbf{X}_j\right)\prod_{k=1}^{M}\times_k U_k^T\right)\end{array} ;(1:M)(1:M)\right\rrbracket\right) \\
&= \underset{U_k^T U_k = I}{\arg\max}\,\mathrm{tr}\left(U_k^T\left(\sum_{i=1}^{N}\sum_{j=1}^{N}\left(b_{ij}-\zeta s_{ij}\right)\left[\begin{array}{l}\mathrm{mat}_k\left(\left(\mathbf{X}_i-\mathbf{X}_j\right)\bar{\times}_k U_k^T\right)\\ \mathrm{mat}_k^T\left(\left(\mathbf{X}_i-\mathbf{X}_j\right)\bar{\times}_k U_k^T\right)\end{array}\right]\right)U_k\right).
\end{aligned}
\tag{3.42}
$$

With (3.42), a solution to the tensorized GEF can be obtained iteratively by the alternating projection method.

GTDA for Human Gait Recognition

A straightforward application of LDA to human gait recognition leads to poor results because of the SSS problem. For this reason, principal component analysis (PCA) [30] is conducted as a preprocessing step to reduce the SSS problem. Unfortunately, some discriminative information is discarded by PCA.

In this Section, we apply the proposed GTDA to reduce the SSS problem in LDA for appearance based human gait recognition [126]. Appearance [83] is the basic stage of visual information representation and reflects the walking manner [45][91]. In the following, we use the averaged gait image [45][91] as the appearance model. As a representation method, the effectiveness of the averaged gait image based recognition has been proved in [45][46][91][92].

The averaged gait image is decomposed by Gabor functions [25][26][101] and we combine the decomposed images to give new representations for recognition. There are three major reasons for introducing the Gabor based representation for the averaged gait image based recognition: 1) human brains seem to have a special function to process information in multi–resolution levels [25][26][101]; 2) it is supposed that Gabor functions are similar to the receptive field profiles in the mammalian cortical simple cells [25][101]; and 3) Gabor function based representations have been successfully employed in many computer vision applications, such as face recognition [87][88][89] and texture analysis [32].

Although Gabor function based representations are effective for object recognition [25][26] and image understanding, the computational cost of the representation is high. Therefore, three variant methods for representation are introduced to utilize Gabor functions to reduce the computational cost in calculating the representation and in training subsequent feature selection algorithms by partially or fully summing over Gabor functions. The sum operation reduces the number of functions, used for decomposing the averaged gait images. These methods are: 1) the sum over directions of Gabor functions based representation (GaborD), 2) the sum over scales of Gabor functions based representation (GaborS), and 3) the sum over both scales and directions of Gabor functions based representation (GaborSD).

Gabor Gait Representation

As demonstrated in [45][91], the averaged gait image is a robust feature for gait recognition tasks. In Figure 3.10, the sample averaged gait images are obtained from different persons under different circumstances. It can be observed that: 1) the averaged gait images of the same person under different circumstances share similar visual effects; and 2) the averaged gait images of different persons, even under the same circumstance, are very different. So, it is possible to recognize a person by his or her averaged gait images. Furthermore, according to research results reported in [80], Gabor functions based image decomposition is biologically relevant to and is useful for image understanding and recognition. Consequently, it is reasonable to introduce Gabor functions for the averaged gait image based gait recognition.

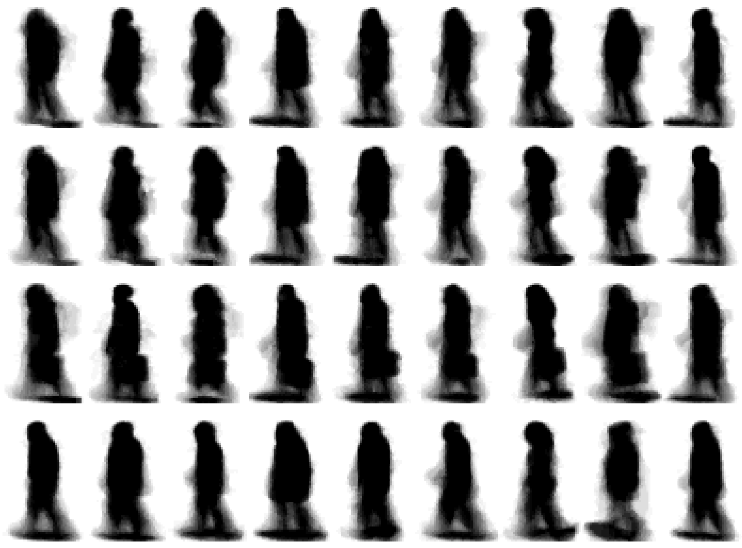


Figure 3.10. The columns show the averaged gait images of nine different people in the Gallery of the USF database described in §1. The four rows in the figure from top to bottom are based on images taken from the Gallery, ProbeB, ProbeH, and ProbeK, respectively. The averaged gait images in a single column come from the same person.

1. Gabor Functions

Marcelja [101] and Daugman [25][26] modelled the responses of the visual cortex by Gabor functions, because they are similar to the receptive field profiles in the mammalian cortical simple cells. Daugman [25][26] developed the 2D Gabor functions, a series of local spatial bandpass filters, which have good spatial localization, orientation selectivity, and frequency selectivity. Lee [80] gave a good introduction to image representation using Gabor functions. A Gabor (wavelet, kernel, or filter) function is the product of an elliptical Gaussian envelope and a complex plane wave, defined as:

$$\psi_{s,d}(x,y) = \psi_{\vec{k}}(\vec{z}) = \frac{\|\vec{k}\|}{\delta^2} \cdot e^{-\frac{\|\vec{k}\|^2 \cdot \|\vec{z}\|^2}{2\delta^2}} \cdot \left[e^{i\vec{k}\cdot\vec{z}} - e^{-\frac{\delta^2}{2}} \right], \tag{3.43}$$

where $\vec{z} = (x,y)$ is the variable in a spatial domain and $\vec{k}$ is the frequency vector, which determines the scale and orientation of Gabor functions, $\vec{k} = k_s e^{i\phi_d}$, where $k_s = k_{\max}/f^s$, $k_{\max} = \pi/2$, $f = 2$, $s = 0,1,2,3,4$, and $\phi_d = \pi d/8$, for $d = 0,1,2,3,4,5,6,7$. Examples of the real part of Gabor functions are presented in Figure 3.11. Here, we use Gabor functions (full complex functions) with five different scales and eight different orientations, making a total of forty Gabor functions, for the averaged gait image decomposition. The number of oscillations under the Gaussian envelope is determined by $\delta = 2\pi$. The term $\exp(-\sigma^2/2)$ is subtracted in order to make the kernel DC–free, and thus insensitive to the average illumination.

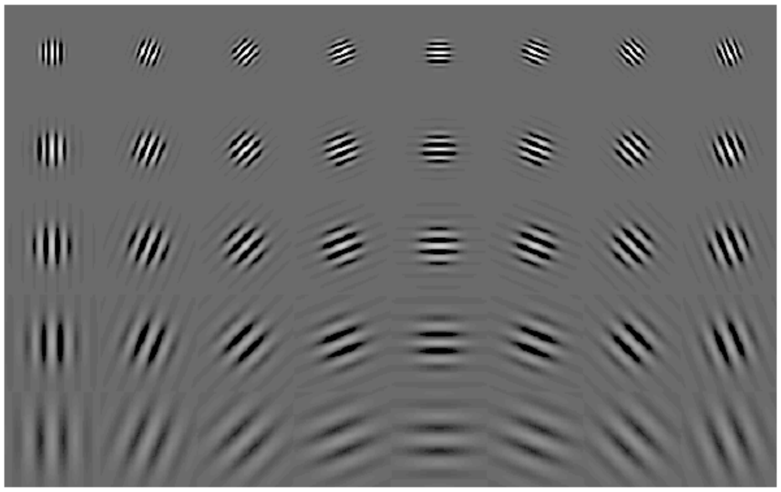


Figure 3.11. The real part of Gabor functions with five different scales and eight different directions.

2. Gabor based Gait Representation

The Gabor function representation of an averaged gait image is obtained by convolving the Gabor functions with the averaged gait image. This yields a fourth order tensor in $R^{L_1 \times L_2 \times 5 \times 8}$ constructed by filtering an averaged gait image through a series of Gabor functions with five scales and eight directions. Two indices are required for pixel locations: one index is required for scale information, and one index is required for direction information. The entries of the fourth order tensor are complex numbers and the magnitude part of this fourth order tensor is defined as the Gabor gait as shown in Figure 3.12. In Gabor gait, there are 40 components (images), each of which is the magnitude part of the output, which is obtained by convoluting the averaged gait image with a Gabor function.

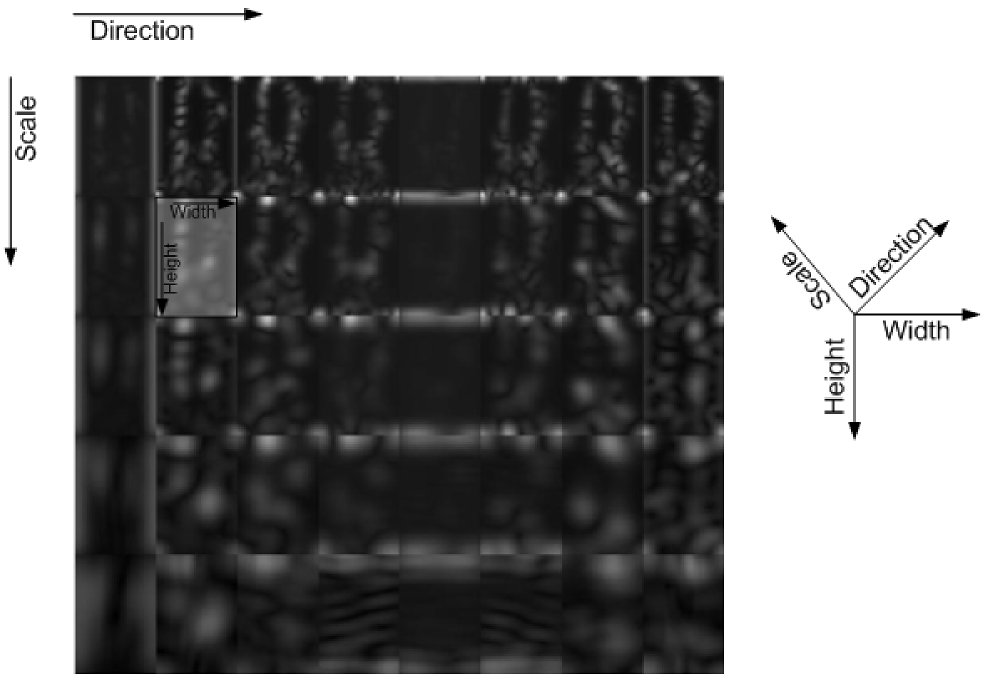


Figure 3.12. Gabor gait: the rows show different scales and the columns show different directions for an averaged gait image.

The gait representation method in Figure 3.12 is similar to the face representation method [87][88] using Gabor functions. Although this method for representation is powerful, its computational costs both for recognition and calculation for representation are higher compared with the original image based recognition. The computational cost in recognition is described in §3.

We introduce three new methods to decompose averaged gait images based on Gabor functions defined in (3.43). These are the sum over directions of Gabor functions based representation (GaborD), the sum over scales of Gabor functions based representation (GaborS), and the sum over scales and directions of Gabor functions based representation (GaborSD). The most important benefit of these new representations is that the cost of computing the gait representation based on them is low. The computational cost of the Gabor based representation and the complexity analysis for GTDA based dimension reduction for different representations are given in §3.

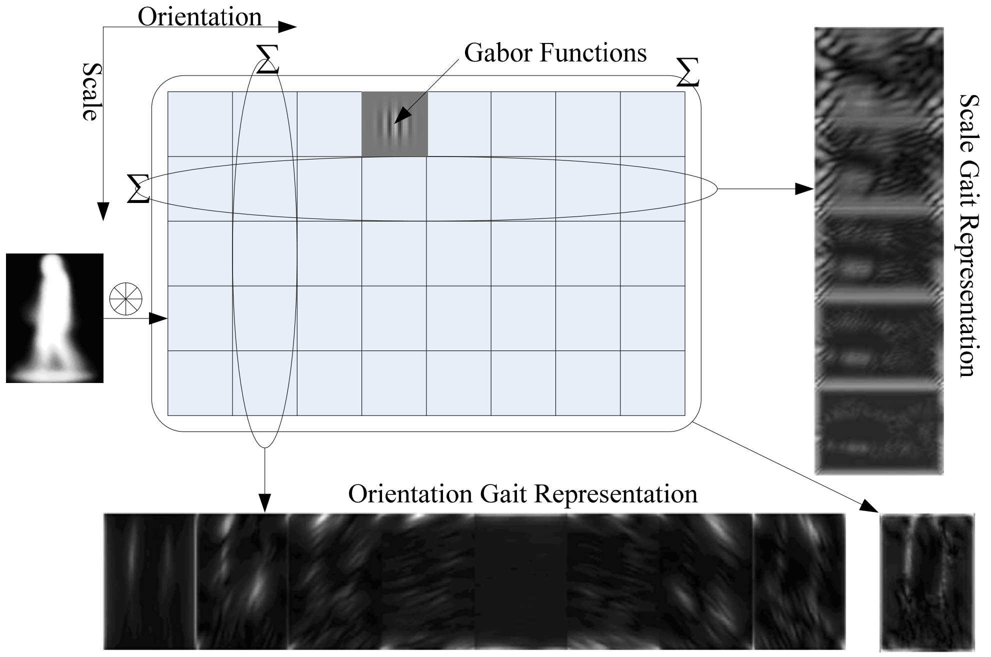


Figure 3.13. Three new methods for averaged gait image representation using Gabor functions: GaborS, GaborD, and GaborSD.

GaborD is the magnitude part of outputs generated by convolving an averaged gait image $I(x,y)$ with the sum of Gabor functions over the eight directions with a fixed scale,

$$\text{GaborD}(x,y) = \left| \sum_d \left(I * \psi_{s,d} \right)(x,y) \right| = \left| \left(I * \sum_d \psi_{s,d} \right)(x,y) \right|, \tag{3.44}$$

where $\psi_{s,d}(x,y)$ is the Gabor function defined in (3.43); and $\text{GaborD}(x,y)$ is the output of the GaborD method for representation. Therefore, we have five different outputs to represent the original gait image through the GaborD decomposition. We then put all $\text{GaborD}(x,y)$ with different scales together as a third order tensor $\mathbf{G}_D$ in $R^{L_1 \times L_2 \times 5}$: two indices are required for pixel locations; and one index is required for scale change. The calculation procedure is shown in Figure 3.13. Examples of GaborD based gait representation are shown in Figure 3.14.

GaborS is the magnitude part of outputs generated by convolving an averaged gait image $I(x,y)$ with the sum of Gabor functions over the five scales with the fixed direction,

$$\text{GaborS}(x,y) = \left| \sum_s \left(I * \psi_{s,d} \right)(x,y) \right| = \left| \left(I * \sum_s \psi_{s,d} \right)(x,y) \right|, \tag{3.45}$$

where $\psi_{s,d}(x,y)$ is the Gabor function defined in (3.43), and $\text{GaborS}(x,y)$ is the output of the GaborS method for representation. Therefore, we have eight different outputs to represent the original gait image through the GaborS based decomposition. We then put all $\text{GaborS}(x,y)$ with different scales together as a third order tensor $\mathbf{G}_S$ in $R^{L_1 \times L_2 \times 8}$: two indices are required for pixel locations; and one index is required for direction change. The calculation procedure is shown in Figure 3.13. Examples of GaborS based gait representation are shown in Figure 3.14.

GaborSD is the magnitude part of the output generated by convolving an averaged gait image $I(x,y)$ with the sum of all forty Gabor functions,

$$\text{GaborSD}(x,y) = \left| \sum_s \sum_d \left(I * \psi_{s,d} \right)(x,y) \right| = \left| \left(I * \sum_s \sum_d \psi_{s,d} \right)(x,y) \right|, \tag{3.46}$$

where $\psi_{s,d}(x,y)$ is the Gabor function defined in (3.43), and $\text{GaborSD}(x,y)$ is the output of the GaborSD method for representation. Therefore, it is a second order tensor in $R^{L_1 \times L_2}$. Two indices are required for pixel locations. The calculation procedure is shown in Figure 3.13. Examples of GaborD based gait representation are shown in Figure 3.14.

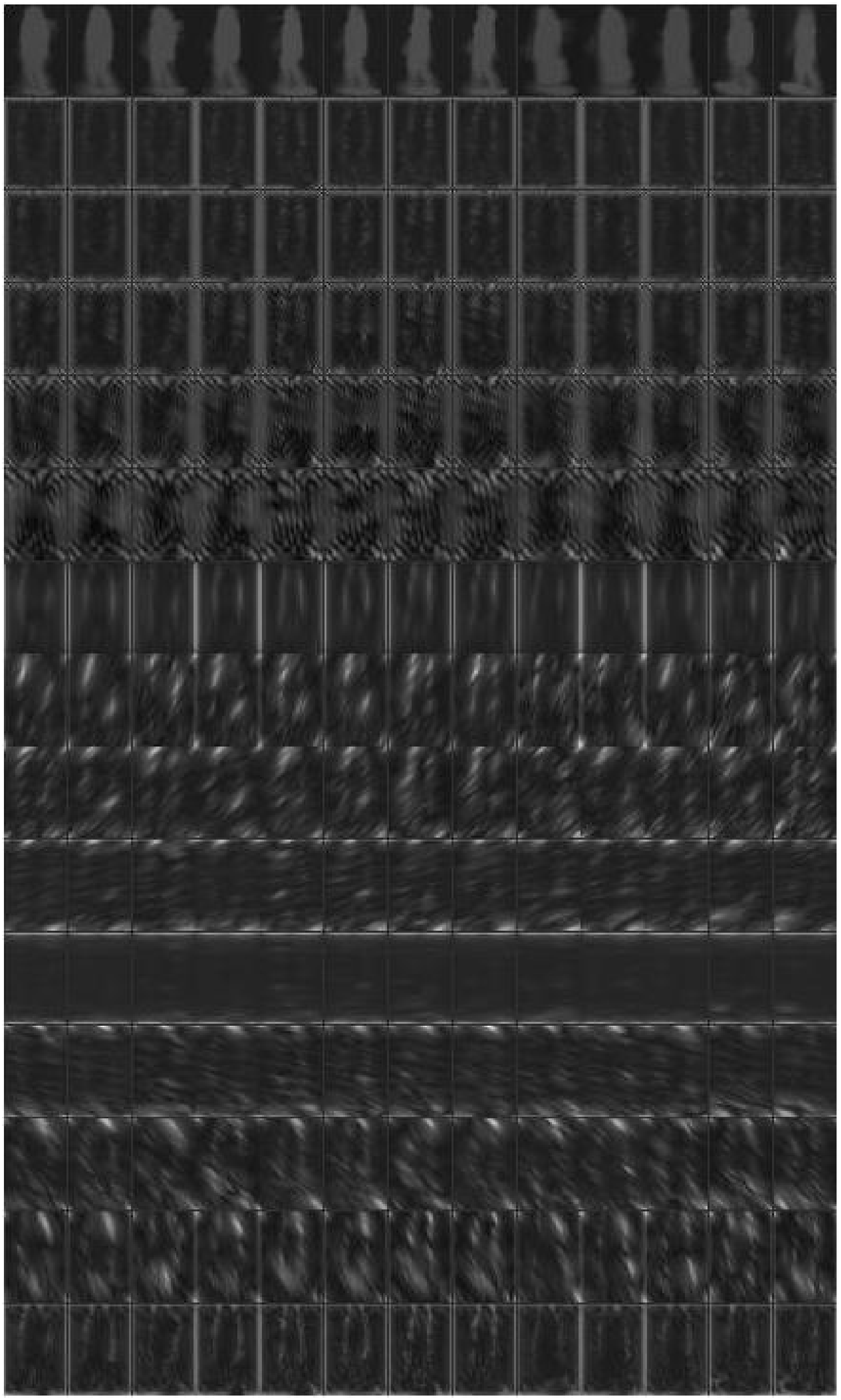


Figure 3.14. The thirteen columns are Gallery gait, ProbeA gait, ProbeB gait, ProbeC gait, ProbeD gait, ProbeE gait, ProbeF gait, ProbeG gait, ProbeH gait, ProbeI gait, ProbeJ gait, ProbeK gait, and ProbeL gait, respectively. From the first row to the last row are the original gait, GaborD (from 0 to 4), GaborS (from 0 to 7), and GaborSD, respectively. The Gallery gait and ProbeA – ProbeI gaits are described in Section 3.8.2.1.

3. Computational Complexity

Gabor functions with different scales and directions are approximated by masks of size $G_1 \times G_2$ (in experiments, $G_1 = G_2 = 64$) and averaged gait images are in $R^{L_1 \times L_2}$. Therefore, the computational complexities for generating a Gabor gait in $R^{L_1 \times L_2 \times 5 \times 8}$, a GaborD gait in $R^{L_1 \times L_2 \times 5}$, a GaborS gait in $R^{L_1 \times L_2 \times 8}$, and a GaborSD gait in $R^{L_1 \times L_2}$ are $O(40 L_1 L_2 G_1 G_2)$, $O(5 L_1 L_2 G_1 G_2)$, $O(8 L_1 L_2 G_1 G_2)$, and $O(L_1 L_2 G_1 G_2)$, respectively. Based on the analysis, the GaborD, GaborS, and GaborSD based gait representation can reduce the computational complexity of the Gabor based representation, because the number of filters (the sum of Gabor functions) for decomposition in the GaborD/GaborS/GaborSD based representation is smaller than the number of filters (Gabor functions) for decomposition in Gabor based representation. The experiments in §2 show that GaborD and GaborS based representations perform slightly better than Gabor based representation for gait recognition.

Table 3.7. Computational complexities of the alternating projection method based optimization procedure of GTDA with Gabor/GaborD/GaborS/GaborSD representations.

	Time Complexity	Space Complexity
Gabor gaits in $R^{L_1 \times L_2 \times 5 \times 8}$	$O\left(T_1\left(637 + L_1^3 + L_2^3\right)\right)$	$O\left(89 + L_1^2 + L_2^2\right)$
GaborD gaits in $R^{L_1 \times L_2 \times 5}$	$O\left(T_2\left(125 + L_1^3 + L_2^3\right)\right)$	$O\left(25 + L_1^2 + L_2^2\right)$
GaborS gaits in $R^{L_1 \times L_2 \times 8}$	$O\left(T_3\left(512 + L_1^3 + L_2^3\right)\right)$	$O\left(64 + L_1^2 + L_2^2\right)$
GaborSD gaits in $R^{N_1 \times N_2}$	$O\left(T_4\left(L_1^3 + L_2^3\right)\right)$	$O\left(L_1^2 + L_2^2\right)$

The computational complexities of the alternating projection method based optimization procedure of GTDA with Gabor/GaborD/GaborS/GaborSD representation are listed in Table 2.1. In Table 2.1, T_1 (T_2, T_3, and T_4) is the number of iterations to make the optimization procedures of GTDA with Gabor (GaborD, GaborS, and GaborSD) based representations converge. In our experiments, we found that T_1, T_2, T_3, and T_4 are usually comparable.

Therefore, with GaborS/GaborD/GaborSD representations, the computational complexities of the alternating projection method based optimization procedure of GTDA are reduced compared with that of the Gabor based representation.

Experimental Results

This Section first briefly describes the USF HumanID gait database [126] (gallery and probes). We then compare the performance of our algorithms with several other established algorithms for human gait recognition.

1. HumanID Gait Database: Gallery and Probe Data Sets

Table 3.8. Twelve probe sets for challenge experiments.

Experiment (Probe)	# of Probe Sets	Difference between Gallery and Probe Set
A (G, A, L, NB, M/N)	122	View
B (G, B, R, NB, M/N)	54	Shoe
C (G, B, L, NB, M/N)	54	View and Shoe
D (C, A, R, NB, M/N)	121	Surface
E (C, B, R, NB, M/N)	60	Surface and Shoe
F (C, A, L, NB, M/N)	121	Surface and View
G (C, B, L, NB, M/N)	60	Surface, Shoe, and View
H (G, A, R, BF, M/N)	120	Briefcase
I (G, B, R, BF, M/N)	60	Briefcase and Shoe
J (G, A, L, BF, M/N)	120	Briefcase and View
K (G, A/B, R, NB, N)	33	Time, Shoe, and Clothing
L (C, A/B, R, NB, N)	33	Time, Shoe, Clothing, and Surface

We carried out all our experiments upon the USF HumanID [126] outdoor gait (people–walking–sequences) database of version 2.1. The database has been built and widely utilized for vision–based gait recognition. It consists of 1,870 sequences from 122 subjects (people). For each subject, there are the following covariates: change in viewpoints (Left or Right), change in shoe types (A or B), change in walking surface (Grass or Concrete), change in carrying conditions (Briefcase or No Briefcase), and elapsed time (May or November) between sequences being compared. There is a set of pre–designed experiments (12 experiments) for algorithm comparison. For algorithm training, the database provides a gallery with the following covariates: grass, shoe type A, right camera, and no briefcase, which was collected in May and it also includes a number of new subjects collected in November. This gallery dataset has 122 individuals. For algorithm testing, 12 probe sets are constructed according to the 12 experiments and detailed information about the probe sets is given in Table 3.8. More detailed information about USF HumanID is described in [126].

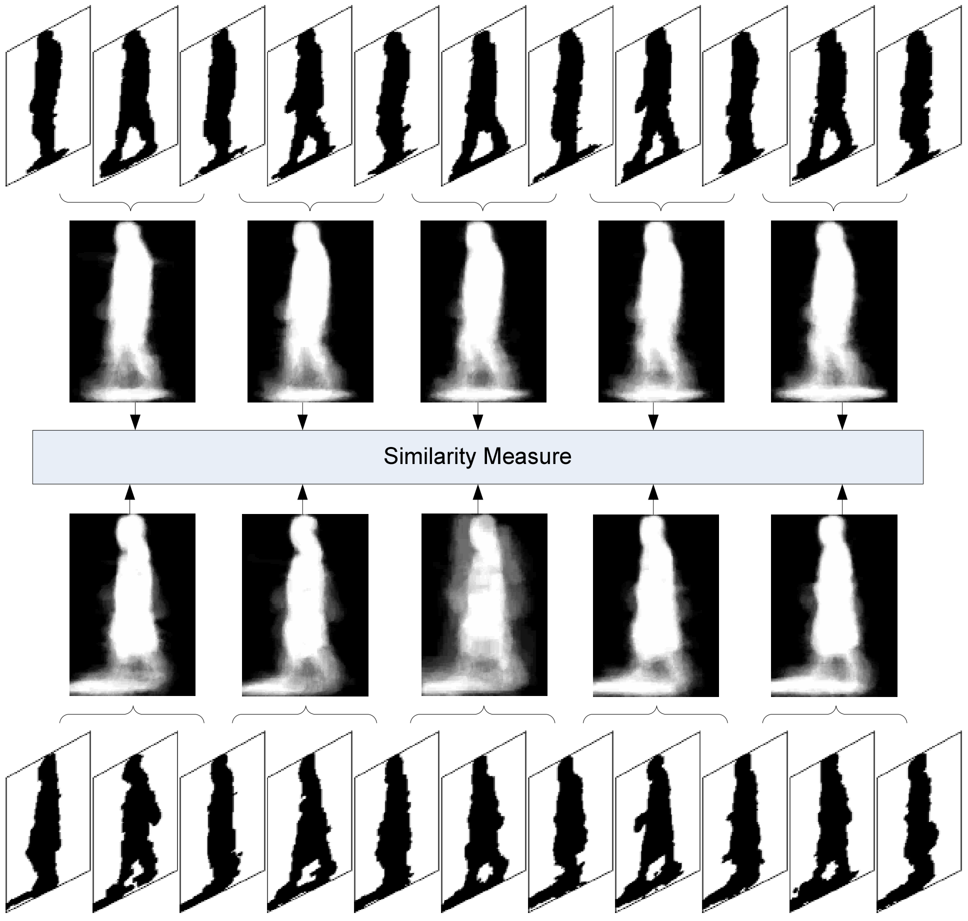


Figure 3.15. The averaged gait extraction and the dissimilarity measure.

Figure 3.15 shows examples of averaged gait images. The averaged gait image stands for the mean image (pixel by pixel) of silhouettes over a gait cycle within a sequence. A gait cycle is two successive half gait cycles. A half gait cycle is a series of stances: from heels–together–stance and full–stride–stance, to heels–together–stance, as shown in Figure 3.15. As suggested in [91], the whole sequence is partitioned into a series of sub–sequences according to the gait period (a cycle) length N_{Gait}. Then the binary images within one cycle (a sub–sequence) are averaged to acquire a set of averaged silhouette images AS_i, i.e.

$$AS_i \big|_{i=1}^{\lfloor T/N_{Gait} \rfloor} = \left(\sum_{k=iN_{Gait}}^{k=(i+1)N_{Gait}-1} S(k) \right) \Big/ N_{Gait} . \tag{3.47}$$

The averaged gait representation is robust against any errors in individual frames, so we choose the averaged gait image to represent a gait cycle. One sequence yields several averaged gait images and the number of averaged gait images depends on the number of gait cycles in this sequence. In the following experiments, averaged gait images are utilized as the original data for the gait recognition problem. Some further averaged gait images from the gallery set are also shown in Figure 3.10, which demonstrate that averaged gait images can be used for gait recognition, because different people have different averaged gait images.

The dissimilarity measure in gait recognition is the same as in [91]. The distance between the gallery sequence and the probe sequence is

$$\begin{aligned} &\mathrm{Dist}\left(AS_P^{Method}, AS_G^{Method}\right) \\ &\quad = \mathrm{Median}_{i=1}^{N_P}\left(\min_{j=1}^{N_G} \left\| AS_P^{Method}(i) - AS_G^{Method}(j) \right\|\right), \end{aligned} \tag{3.48}$$

where $AS_P^{Method}(i)\big|_{i=1}^{N_P}$ is the i^{th} projected AS in the probe data and $AS_G^{Method}(j)\big|_{j=1}^{N_G}$ is the j^{th} projected AS in the gallery. The right hand side of (3.48) is the median of the Euclidean distances between averaged silhouettes from the probe and the gallery. It is suggested as a suitable measure for gait recognition by Liu and Sarkar in [91].

There are only two parameters in all proposed methods, one for GTDA and the other for LDA. In detail, one parameter is the threshold value δ for GTDA as described in (3.37). In all experiments, we vary δ from 0.85 to 0.995 with a step 0.005. In 2DLDA, a similar strategy is used, i.e., δ is used to determine the dimensions of the projected subspace in each order. The other parameter is the number of selected dimensions in LDA. In all experiments relevant to LDA, we vary the number of dimensions from 1 to 121 with a step 1. To speed up all experiments, we down sample the original averaged gait images from 128×88 to 64×44 in all proposed methods. These are indicated by the note “H” in Table 3.9

and Table 3.10. We also show some experimental results based on the original averaged gait images with the size 128×88.

To examine the effectiveness of the automatic selection of ζ for GTDA based recognition defined in (3.31), we also manually tune the parameters to achieve further improvement by changing the selected dimensions for each mode and the Lagrange multiplier ζ defined in (3.36). This is indicated by the note "M" in Table 3.9 and Table 3.10. Although manually tuning parameters improves the performance, it is time consuming. Moreover, the improvement is limited, so the automatic parameter selection used in this Chapter is enough for applications.

2. Performance Evaluation

Sarkar et al. [126] evaluated the performance of the baseline algorithm on the HumanID challenge database using the rank one/five recognition rates: 1) the rank one recognition rate is the percentage of the number of the correct subjects in the first place of a list of matches obtained by an algorithm and 2) the rank five recognition rate is the percentage of the number of the correct subjects in any of the first five places of a list of matches obtained by an algorithm. Twelve experiments have been designed, namely experiment A to experiment L, as shown in Table 3.8. The baseline algorithm reports the rank one recognition rates of the twelve experiments with increasing difficulty from 78% as the easiest to 3% as the hardest by examining the effects of the introduced five covariates (under different combinations).

Table 3.9 and Table 3.10 report all experiments, which compare the proposed algorithms with the existing algorithms. The item "Avg" in Table 3.9 and Table 3.10 means the averaged recognition rates of all probes (A–L), i.e., the ratio of correctly recognized subjects to the total number of subjects in all probes. The columns labeled A to L are exactly the same tasks as in the baseline algorithm. In both tables, the first rows give the performance of Baseline [126], HMM [66], IMED [167], IMED+LDA [167], LDA [45], LDA+Sync [45], LDA+Fusion [45], 2DLDA [173], and 2DLDA+LDA [173], respectively; while the performance of the new algorithms upon the same gallery set and probe set is fully reported on all comparison experiments, which are namely, GTDA(H), GTDA (M & H), GTDA, Gabor+GTDA(H), GaborD+GTDA(H), GaborD+GTDA, GaborS+GTDA (H), GaborS+GTDA, GaborSD+GTDA(H), GaborSD+GTDA, GTDA+LDA(H), GTDA+LDA, Gabor+GTDA+LDA(H), GaborD+GTDA+LDA(H), GaborD+ GTDA+LDA, GaborS+GTDA+LDA(H), GaborS+GTDA+LDA, and GaborSD+ GTDA+LDA(H), respectively. Finally, the last columns of both tables report the average performance of the corresponding algorithms on all probe sets.

From the comparison results in Table 3.9 and Table 3.10, it is clear that the averaged recognition rate of the twelve probes, our new methods (GTDA+LDA, Gabor+GTDA+LDA, GaborD+GTDA+LDA, and GaborS+GTDA+LDA) outperform the previous state–of–the–art algorithms (top part in both tables), e.g., the HMM algorithm, which is stable in modeling the gait cycles, and the IMED algorithm, which is demonstrated to improve the conventional LDA. Figure 3.16 and Figure 3.17 visually compare the results obtained from some important algorithms with the results obtained from the proposed ones. Table 3.9 and Table 3.10 show that our proposed methods are not very sensitive to the changes in the size of averaged gait images, because the recognition rates are only slightly decreased when averaged gait images are down sampled from 128×88 to 64×44. Manually tuning the parameter ζ in GTDA in (15) will slightly improve the averaged recognition rate. Furthermore, performances for probes D–G and K–L are not satisfactory. Therefore, further studies are required to make them applicable. Finally, the performances of different methods have the following relationship: Baseline < IMED < LDA < IMED+ LDA < 2DLDA+LDA < LDA+Fusion < GTDA+LDA < GaborD+GTDA+LDA < Gabor+GTDA+LDA < GaborS+GTDA+LDA.

In addition, it is worth emphasizing that the effects of the five covariates are also reflected in experimental results. In general, the results in Table 3.9 and Table 3.10, for the proposed GaborS/GaborD+GTDA+LDA, show that:

- *Viewpoint* and *shoe* changes have little impact on the recognition rate. This point is demonstrated by columns A–C, in which the rank one recognition rates are around 92%;
- Apart from the *viewpoint* and the *shoe* covariates, if *briefcase* is also considered, the recognition tasks become more difficult and as a result, in columns H–J the performance is around 87% in rank one evaluation;

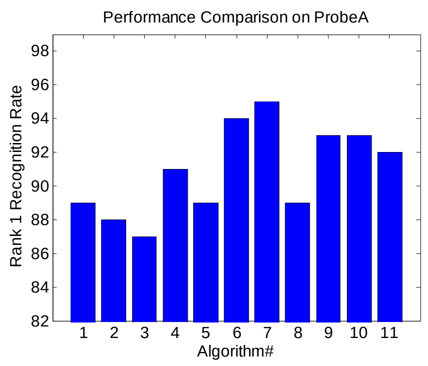
- If the *viewpoint* and the *shoe* issues are studied together with the *surface* covariate, the recognition tasks become hard. This effect leads to a worse performance around 35% in columns D–G in rank one evaluation;
- The most difficult problem in human gait recognition is the *elapsed* time task, i.e., data in gallery and data in probes were captured at different time. In USF HumanID, data in gallery were obtained in May and data in probes were obtained in November. Much work should be done to improve the performance on the tasks K and L although our proposed algorithms report better performance around 17% in rank one evaluation compared with many previous efforts, such as the baseline [126], IMED [167], IMED+LDA, LDA [45][46], and LDA+Fusion [45][46], in most cases.

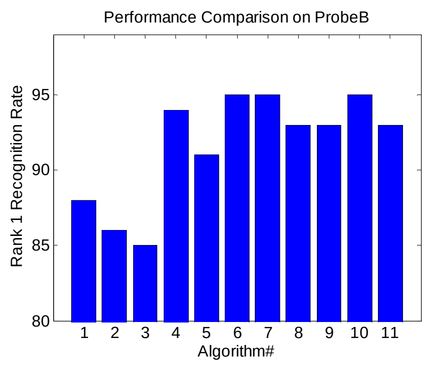
Table 3.9. Rank one recognition rates for human gait recognition.

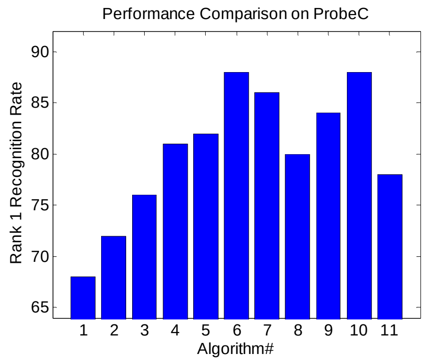
Rank One (%)	A	B	C	D	E	F	G	H	I	J	K	L	Avg
Probe Size	122	54	54	121	60	121	60	120	60	120	33	33	—
Baseline	73	78	48	32	22	17	17	61	57	36	3	3	40.9572
HMM	89	88	68	35	28	15	21	85	80	58	17	15	53.5365
IMED	75	83	65	25	28	19	16	58	60	42	2	9	42.8695
IMED+LDA	88	86	72	29	33	23	32	54	62	52	8	13	48.6357
LDA	87	85	76	31	30	18	21	63	59	54	3	6	48.1983
LDA+Sync	83	94	61	50	48	22	33	48	52	34	18	12	48.0355
LDA+Fusion	91	94	81	51	57	25	29	62	60	57	9	12	55.8257
2DLDA	89	93	80	28	33	17	19	74	71	49	16	16	50.9823
2DLDA+LDA	89	91	82	33	33	23	25	67	78	50	19	19	52.6409
GTDA (H)	85	88	73	24	25	15	14	53	49	45	4	7	42.9916
GTDA (M & H)	86	88	73	24	25	17	16	53	49	45	10	7	43.7035
GTDA	85	88	71	19	23	15	14	49	47	45	7	7	41.5992
Gabor+GTDA (H)	84	86	73	31	30	16	18	85	85	57	13	10	52.5052
GaborD+GTDA (H)	88	88	71	28	28	12	19	87	75	59	7	10	51.7359
GaborD+GTDA	81	88	65	21	23	8	13	92	83	55	13	10	49.2610
GaborS+GTDA (H)	89	89	69	31	33	13	16	79	76	56	13	13	51.4322
GaborS+GTDA	82	86	67	22	30	8	14	92	88	62	10	7	50.9990
GaborSD+GTDA (H)	87	89	71	23	28	8	14	82	69	51	4	13	48.2109
GaborSD+GTDA	81	82	69	17	26	7	14	91	78	60	10	10	48.8518
GTDA+LDA (H)	94	95	88	35	42	23	30	65	61	58	16	19	54.5543
GTDA+LDA	95	95	86	39	44	25	30	61	68	67	16	19	56.5167
Gabor+GTDA+LDA (H)	89	93	80	45	49	23	30	81	85	53	10	19	57.7296
GaborD+GTDA+LDA (H)	93	93	84	34	40	23	32	90	80	63	16	19	58.9102
GaborD+GTDA+LDA	89	93	84	27	35	17	26	93	88	67	16	22	57.5511
GaborS+GTDA+LDA (H)	93	95	88	39	47	28	33	82	82	63	19	19	60.2390
GaborS+GTDA+LDA	91	93	86	32	47	21	32	95	90	68	16	19	60.5804
GaborSD+GTDA+LDA (H)	92	93	78	30	38	21	26	82	75	55	16	19	54.8685

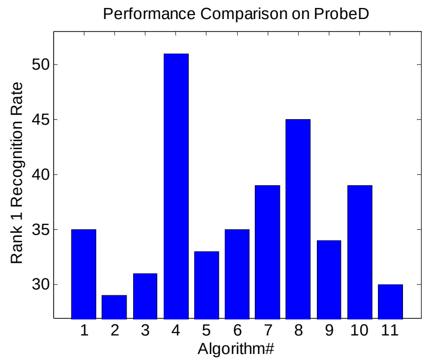
Table 3.10. Rank five recognition rates for human gait recognition.

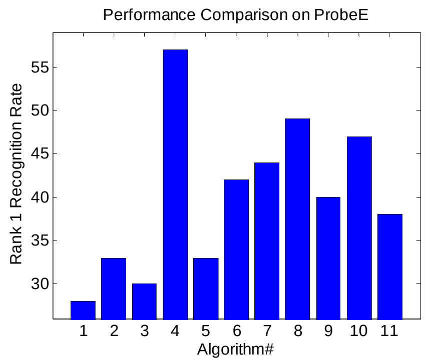
Rank Five (%)	A	B	C	D	E	F	G	H	I	J	K	L	Avg
Probe Size	122	54	54	121	60	121	60	120	60	120	33	33	—
Baseline	88	93	78	66	55	42	38	85	78	62	12	15	64.5397
HMM	—	—	—	—	—	—	—	—	—	—	—	—	—
IMED	91	93	83	52	59	41	38	86	76	76	12	15	65.3132
IMED+LDA	95	95	90	52	63	42	47	86	86	78	21	19	68.5950
LDA	92	93	89	58	60	36	43	90	81	79	12	12	67.3674
LDA+Sync	92	96	91	68	69	50	55	80	78	69	39	30	70.8528
LDA+Fusion	94	96	93	85	79	52	57	89	86	77	24	21	76.1754
2DLDA	97	93	93	57	59	39	47	91	94	75	37	34	70.9530
2DLDA+LDA	97	100	95	58	57	50	50	86	94	77	43	40	72.8507
GTDA (H)	98	95	95	57	54	34	42	75	80	69	22	16	65.0532
GTDA (M & H)	100	97	95	57	54	34	45	75	80	70	25	25	66.1472
GTDA	100	97	95	52	52	34	45	47	71	70	25	25	64.7015
Gabor+GTDA (H)	96	95	89	59	63	33	49	94	92	76	19	40	70.3205
GaborD+GTDA (H)	96	95	88	59	49	27	35	95	97	84	28	28	69.0898
GaborD+GTDA	96	91	82	45	45	23	32	96	94	78	31	37	65.4134
GaborS+GTDA (H)	98	97	93	60	52	34	37	93	95	79	31	25	70.0605
GaborS+GTDA	96	91	84	45	54	23	37	96	95	79	22	31	66.0741
GaborSD+GTDA (H)	95	93	88	54	47	27	30	89	88	71	28	28	64.8361
GaborSD+GTDA	96	91	82	43	54	23	33	98	94	82	28	34	66.3319
GTDA+LDA (H)	100	99	97	66	68	50	57	89	85	81	40	31	75.3267
GTDA+LDA	100	99	97	67	69	50	57	90	90	84	40	37	76.5365
Gabor+GTDA+LDA (H)	95	97	93	70	71	44	56	94	95	80	31	34	75.1451
GaborD+GTDA+LDA (H)	98	99	95	62	68	44	50	96	99	87	37	43	76.0731
GaborD+GTDA+LDA	98	99	93	52	59	37	49	99	99	88	34	43	73.5846
GaborS+GTDA+LDA (H)	98	99	97	68	68	50	56	95	99	84	40	40	77.5762
GaborS+GTDA+LDA	98	99	95	58	64	41	52	98	99	87	31	37	74.9008
GaborSD+GTDA+LDA (H)	99	99	93	57	61	40	47	89	90	78	40	37	71.6534

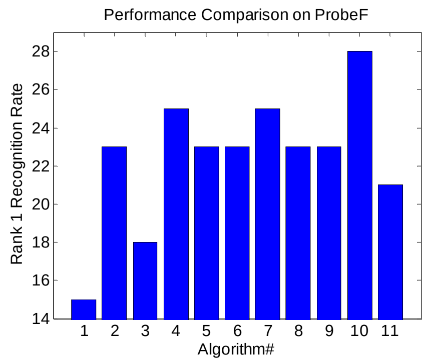

Performance Comparison on ProbeA
Rank 1 Recognition Rate
Algorithm#

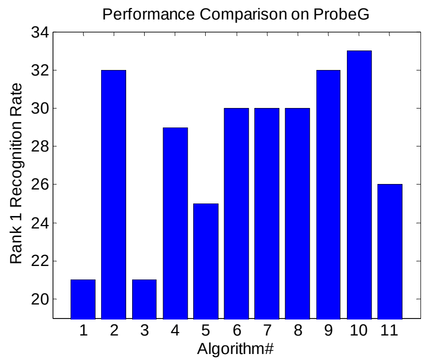

Performance Comparison on ProbeB
Rank 1 Recognition Rate
Algorithm#

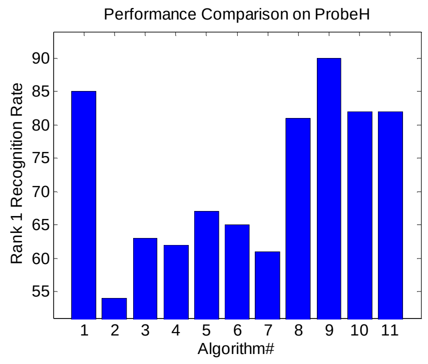

Performance Comparison on ProbeC
Rank 1 Recognition Rate
Algorithm#


Performance Comparison on ProbeD
Rank 1 Recognition Rate
Algorithm#


Performance Comparison on ProbeE
Rank 1 Recognition Rate
Algorithm#


Performance Comparison on ProbeF
Rank 1 Recognition Rate
Algorithm#


Performance Comparison on ProbeG
Rank 1 Recognition Rate
Algorithm#


Performance Comparison on ProbeH
Rank 1 Recognition Rate
Algorithm#

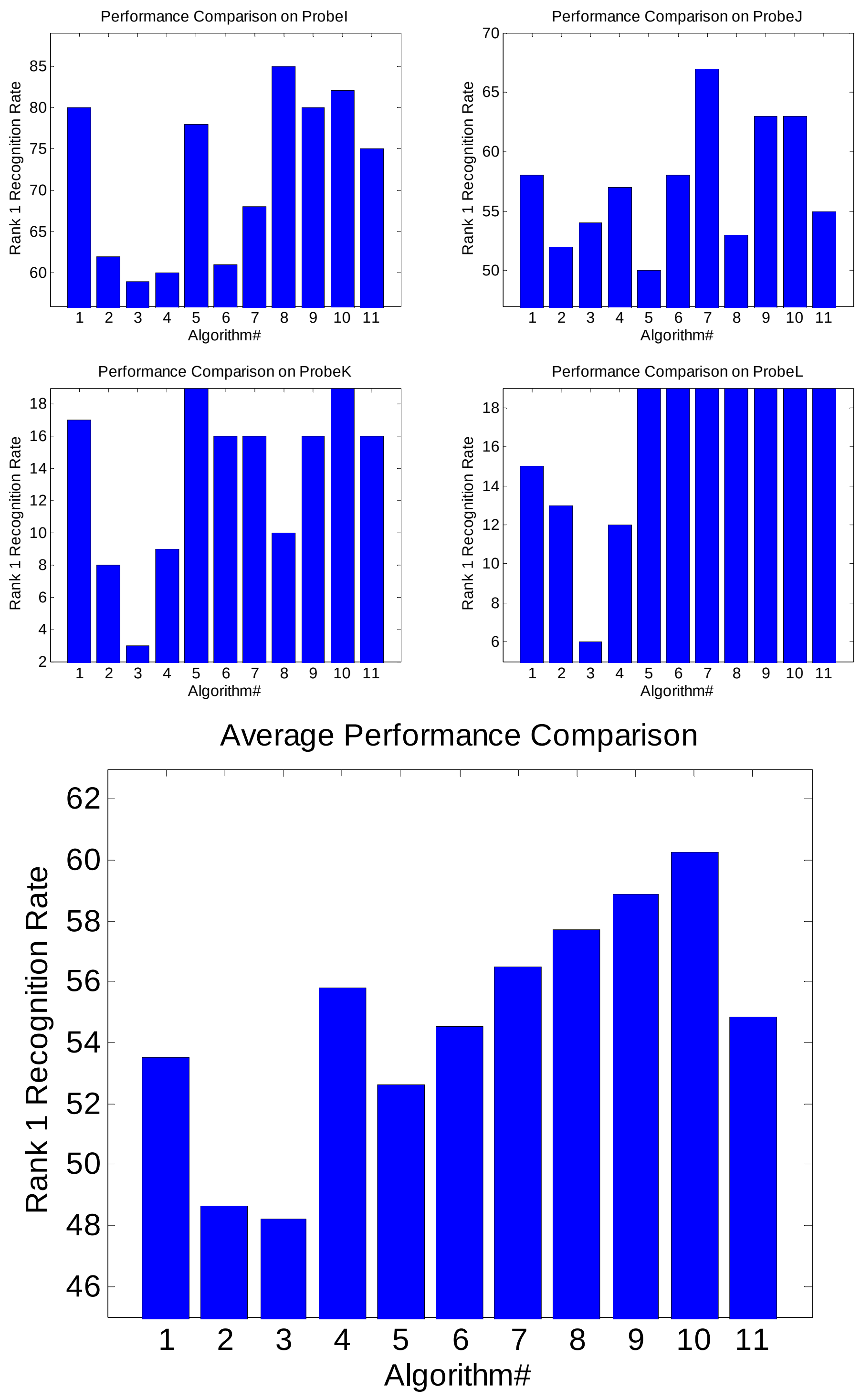
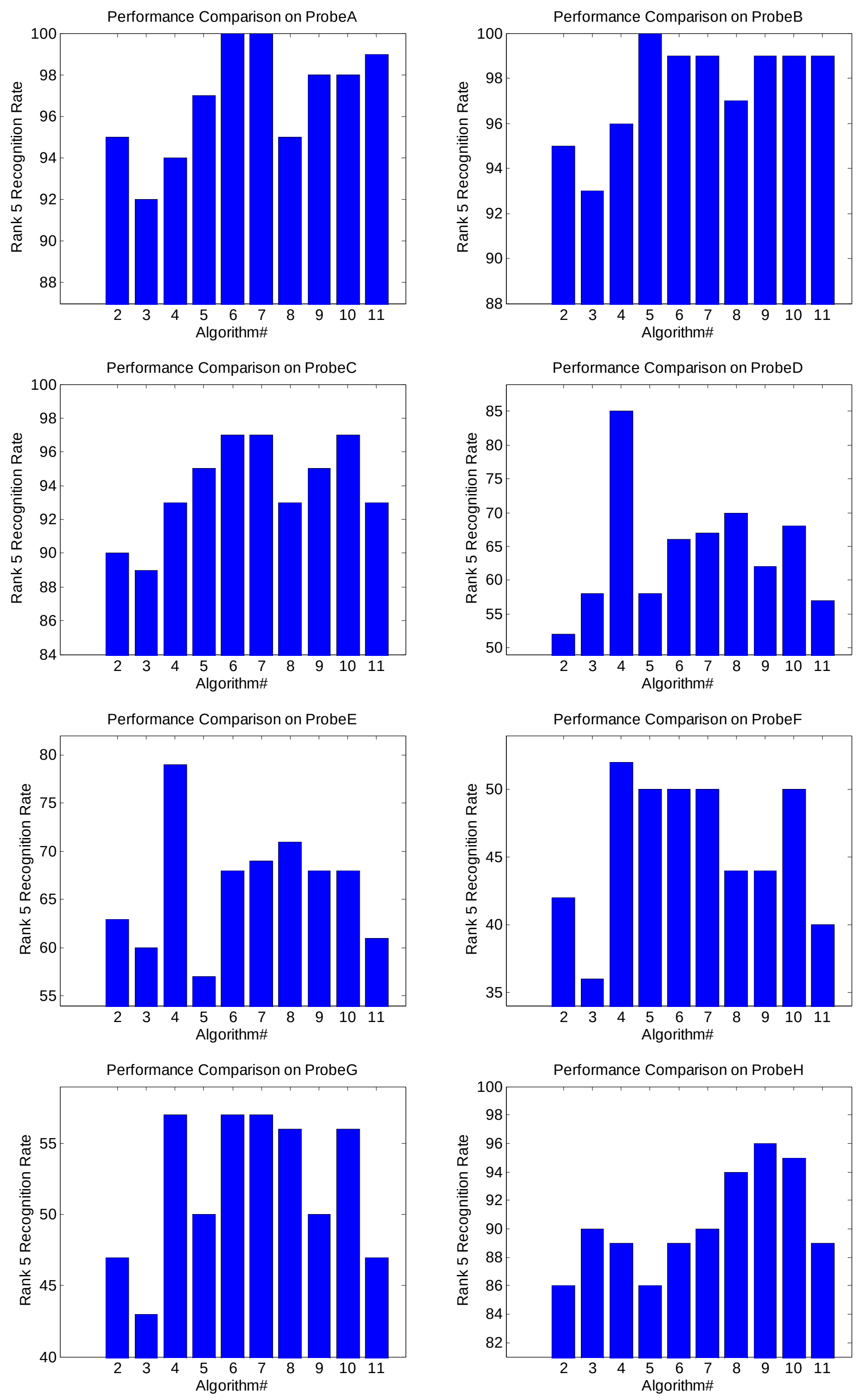


Figure 3.16. Recognition performance comparison for rank one evaluation. From top–left to bottom, in each of the thirteen subfigures (Probes A, B, C, D, E, F, G, H, I, J, K, L, and the average performance), there are eleven bars, which correspond to the performance of HMM, IMED+LDA, LDA, LDA+Fusion, 2DLDA+LDA, GTDA+LDA(H), GTDA+LDA, Gabor+GTDA+LDA(H),

GaborD+GTDA+LDA(H), GaborS+GTDA+LDA(H), and GaborSD+GTDA+ LDA(H), respectively.



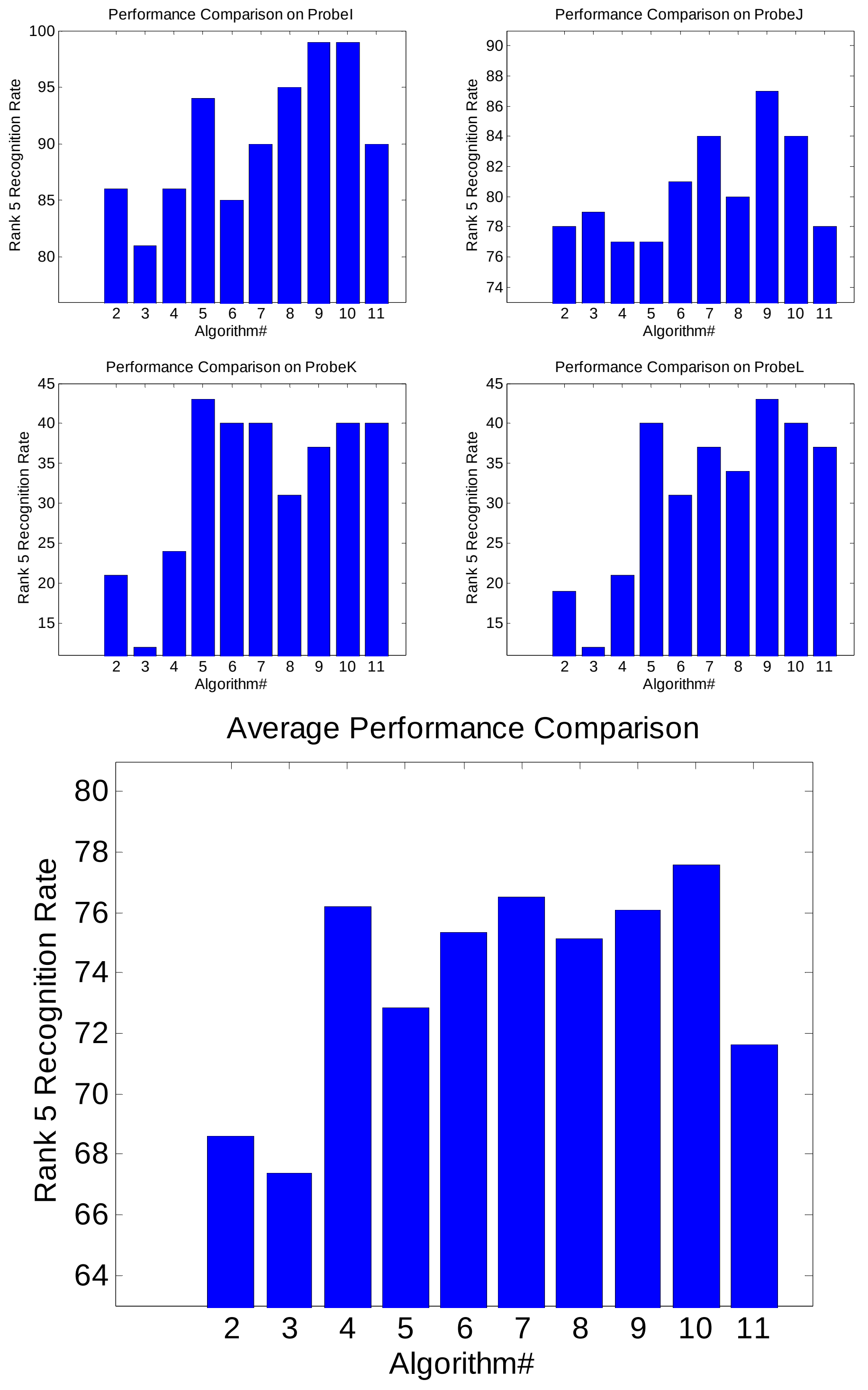


Figure 3.17. Recognition performance comparison for rank five evaluation. From top–left to bottom, in each of the thirteen subfigures (Probes A, B, C, D, E, F, G, H, I, J, K, L, and the average performance), there are ten bars, which correspond to the performance of IMED+LDA, LDA, LDA+Fusion, 2DLDA+LDA, GTDA +LDA(H), GTDA+LDA, Gabor+GTDA+LDA(H), GaborD+GTDA+LDA(H), GaborS+GTDA+LDA(H), and GaborSD+GTDA+LDA(H), respectively.

3. Convergence Examination

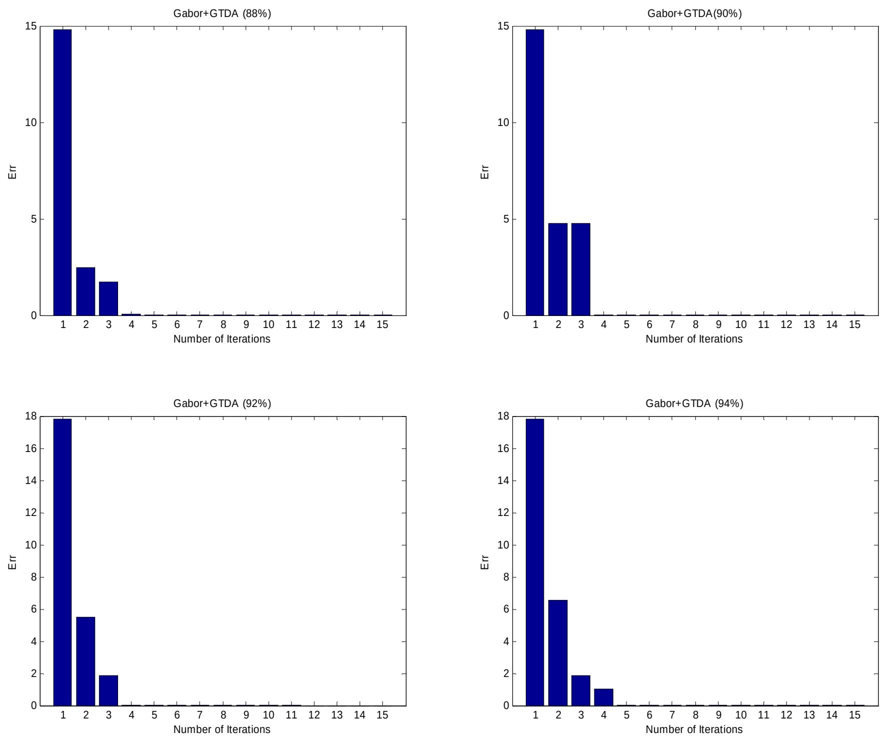


Figure 3.18. Experimental based convergence justification for the alternating projection method for GTDA. The x–coordinate is the number of training iterations and the y–coordinate is the error value Err, as defined in Step 6 in Table 3.6. From left to right, these four sub–figures show how Err changes with the increasing number of training iterations with different threshold values (88%, 90% 92% and 94%) defined in (3.37).

From Figure 3.18, it can be seen that only 3 to 5 iterations are usually required to achieve convergence of the alternating projection method based optimization procedure of GTDA because errors with different threshold values σ approach zero rapidly. In contrast, the traditional 2DLDA does not converge during the training procedure, as shown in the first figure in [173].

Summary

Objects in computer vision research are naturally represented by general tensors. The most popular examples are images and video shots, e.g., face images in face recognition and video shots in video categorization. However, tensors have in the past been reduced to the vector form, because available subspace selection methods do not accept tensors as input. The vectorization removes the structure information, which could reduce the number of parameters needed to model the samples (images and video shots). To preserve the structure information, we develop the general tensor discriminant analysis (GTDA) for discriminative multilinear susbapce selection. This is an effective and efficient preprocessing step for subsequent classification, e.g. by linear discriminant analysis (LDA). Compared with existing multilinear subspace selection methods, e.g., the general tensor analysis (GTA) and the two dimensional LDA (2DLDA), the advantages of the proposed GTDA are: 1) the proposed alternating projection method to obtain a solution of GTDA converges; 2) GTDA accepts general tensors as input; 3) GTDA takes the discriminative information into account; and 4) GTDA reduces the SSS problem of the subsequent classification, e.g., by LDA. We further develop the manifold learning using tensor representations, which is an extension of GTDA based on the graph embedding framework. With this new framework, most of the popular manifold learning algorithms accept tensors as input. Finally, we apply GTDA to human gait recognition and achieve top level performance.

4. Supervised Tensor Learning

In vector based learning[15] [30][39], a projection vector $\vec{w} \in R^L$ and a bias $b \in R$ are learnt to determine the class label of a sample $\vec{x} \in R^L$ according to a linear decision function $y(\vec{x}) = \text{sign}\left[\vec{w}^T \vec{x} + b\right]$. The $\vec{w}$ and b are obtained based on a learning model, e.g., minimax probability machine (MPM) [74][135], based on N training samples associated with labels $\left\{\vec{x}_i \in R^L, y_i\right\}$, where y_i is the class label, $y_i \in \{+1, -1\}$, and $1 \le i \le N$. In this Chapter, we focus on the convex optimization based learning, which accept vectors as input.

Supervised tensor learning (STL) [149][150] is developed to extend the vector based learning algorithms to accept tensors as input. That is we learn a series of projection vectors $\vec{w}_k \mid_{k=1}^{M} \in R^{L_k}$ and a bias $b \in R$ to determine the class label $\{+1, -1\}$ of a sample $\mathbf{X} \in R^{L_1 \times L_2 \times \cdots L_M}$ according to a multilinear decision function $y(\mathbf{X}) = \text{sign}\left[\mathbf{X} \prod_{k=1}^{M} \times_k \vec{w}_k + b\right]$. The $\vec{w}_k \mid_{k=1}^{M}$ and b are obtained from a learning model, e.g., tensor minimax probability machine (TMPM), based on N training samples associated with labels $\left\{\mathbf{X}_i \in R^{L_1 \times L_2 \times \cdots L_M}, y_i\right\}$, where y_i is the class label, $y_i \in \{+1, -1\}$, and $1 \le i \le N$.

This extension to tensor input is important, because many objects in computer vision research are represented by general tensors in $R^{L_1 \times L_2 \times \cdots L_M}$, as described in Chapter 3. If we choose to use vector based learning algorithms, the vectorization operation $\text{vect}(\cdot)$ is applied to a general tensor $\mathbf{X}$ and form a vector $\vec{x} = \text{vect}(\mathbf{X}) \in R^L$, where $L = L_1 \times L_2 \times \cdots L_M$. The vectorization eliminates the structure information of a sample in its original format. However, the information is helpful to reduce the number of parameters in a learning model and result in alleviating the overfitting problem. Usually, the testing error decreases with respect to the increasing complexity of training samples. When the complexity of training samples (partially represented by the number of training samples) is

[15] We refer to the binary classification tasks.

limited, the tensor based learning machine performs better than the vector based learning machine. Otherwise, the vector based learning machine outperforms the tensor based learning machine, as shown in Figure 4.1. Usually, the size of the training set measures the data complexity and the complexity of a suitable classifier should consist with the complexity of the training data.

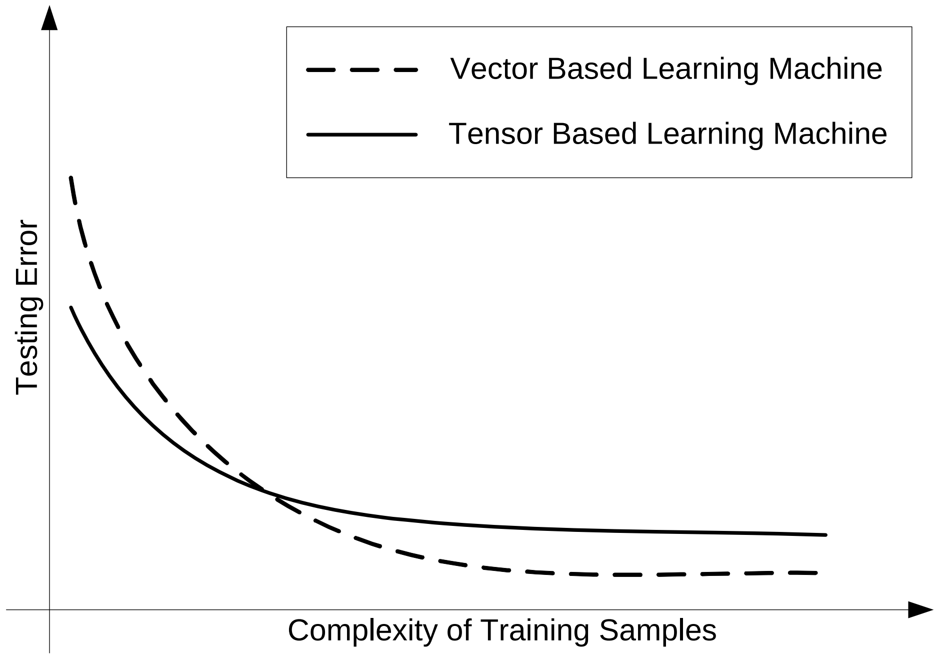


Figure 4.1. Tensor based learning machine vs. vector based learning machine.

This Chapter is organized as follows. In §0, the convex optimization is briefly reviewed and a framework level formula of the convex optimization based learning is introduced. In §0, we develop a supervised tensor learning (STL) framewok, which is an extension of the convex optimization based learning. An alternating projection method is also developed to obtain the solution to an STL based learning algorithm. In §0, we develop a number of tensor extensions of many popular learning machines, such as the support vector machine (SVM) [15][128][130][137][138][161], the minimax probability machine (MPM) [74] [135], the Fisher discriminant analysis (FDA) [37][30][69], and the distance metric learning (DML) [169]. In §169, an iterative feature extraction model (IFEM) is given as an extension of the STL framework and the tensor rank one discriminant analysis (TR1DA) is developed as an example. In §4.5, Two experiments are conducted to study the effectiveness of TMPM (for STL) and

TR1DA (for IFEM) empirically. The first experiment, for image classification, demonstrates that TMPM reduces the overfitting problem in MPM. The second experiment, for the elapsed time problem in human gait recognition, shows TR1DA is more effective than PCA, LDA, and TR1A. Frinally, we summarize this Chapter in §169.

Convex Optimization based Learning

Learning models are always formulated as optimization problems [170][178]. Therefore, mathematical programming [170][178] is the heart of machine learning research [128]. In this Section, we first introduce the fundamentals of convex optimization and then give a general formulation for convex optimization based learning.

A mathematical programming problem [170][178][11] has the form or it can be transformed to this form

$$\begin{bmatrix} \min\limits_{\vec{w}} & f_0(\vec{w}) & \\ \text{s.t.} & \begin{array}{l} f_i(\vec{w}) \le 0, \quad 1 \le i \le m \\ h_i(\vec{w}) = 0, \quad 1 \le i \le p \end{array} \end{bmatrix}, \tag{4.1}$$

where $\vec{w} = [w_1, w_2, \cdots, w_n]^T \in R^n$ is the optimization variable in (4.1); the function $f_0 : R^n \to R$ is the objective function; the functions $f_i : R^n \to R$ are inequality constraint functions; and the functions $h_i : R^n \to R$ are equality constraint functions. A vector $\vec{w}*$ is a solution to the problem if f_0 achieves its minimum among all possible vectors, i.e., all vectors which satisfy the constraint equations ($f_i |_{i=1}^m$ and $h_i |_{i=1}^p$).

When the objective function $f_0(\vec{w})$ and the inequality constraint functions $f_i(\vec{w})|_{i=1}^m$ satisfy

$$\begin{array}{l} f_i(\alpha\vec{w}_1 + \beta\vec{w}_2) \le \alpha f_i(\vec{w}_1) + \beta f_i(\vec{w}_2) \\ \alpha, \beta \in R_+ \quad \text{and} \quad \alpha + \beta = 1 \\ w_1, \vec{w}_2 \in R^n \end{array} \tag{4.2}$$

(i.e., $f_i(\vec{w})|_{i=0}^m$ are convex functions) and the equality constraint functions $h_i(\vec{w})|_{i=1}^p$ are affine (i.e., $h_i(\vec{w}) = 0$ can be simplified as $\vec{a}_i^T \vec{w} = b_i$), the mathematical programming problem defined in (4.1) is named the convex optimization problem. Therefore, a convex optimization problem [11] is defined by

$$\begin{bmatrix} \min\limits_{\vec{w}} & f_0(\vec{w}) & \\ \text{s.t.} & f_i(\vec{w}) \le 0, & 1 \le i \le m \\ & \vec{a}_i^T \vec{w} = b_i, & 1 \le i \le p \end{bmatrix}, \tag{4.3}$$

where $f_i(\vec{w})|_{i=0}^m$ are convex functions. The domain D of the problem in (4.3) is the intersection of the domains of $f_i(\vec{w})|_{i=0}^m$, i.e., $D = \bigcap_{i=0}^{m} \operatorname{dom} f_i$. The point $\vec{w}^*$ in D is the optimal solution of (4.3) if and only if

$$\nabla^T f_0(\vec{w}^*)(\vec{w} - \vec{w}^*) \ge 0, \quad \forall \vec{w} \in D. \tag{4.4}$$

The geometric interpretation of the optimal solution for a convex optimization problem is given in Figure 4.2.

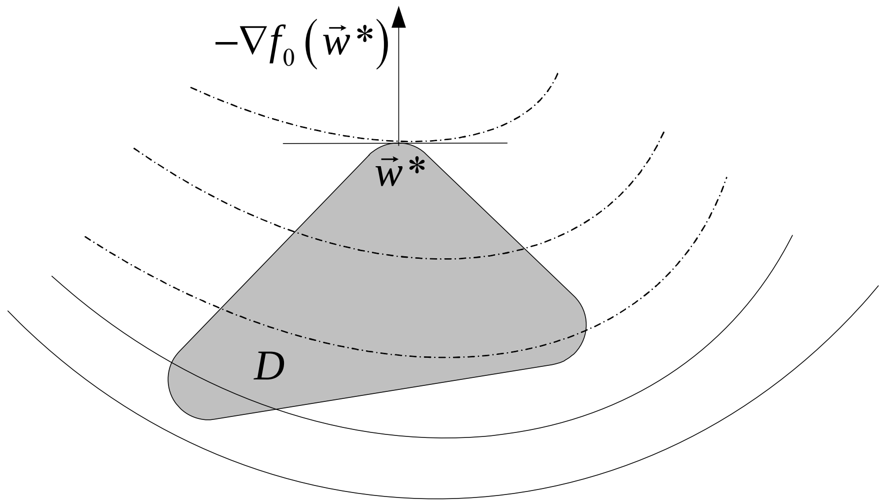


Figure 4.2. The geometric interpretation of the optimal solution $\vec{w}^*$ in D for a convex optimization problem defined in (4.3).

The convex optimization problem defined in (4.3) has a large number of popular special cases, such as linear programming (LP) [160], linear fractional programming (LFP) [11], quadratic programming (QP) [114], quadratically constrained quadratic programming (QCQP) [93], second order cone programming (SOCP) [93], semi–definite programming (SDP) [159], and geometric programming (GP) [12]. All of these special cases have been widely applied in different areas, such as computer networks, machine learning, computer vision, psychology, health research, automation research, and economics.

The significance of a convex optimization problem is that the solution is unique (i.e., the locally optimal solution is also the globally optimal solution), so the convex optimization has been widely applied to machine learning for many years, such as LP [160] in the linear programming machine (LPM) [117][134], QP [114] in the support vector machine (SVM) [161][15][130][128][137][138], SDP [159] in the distance metric learning (DML) [169] and the kernel matrix learning [73], and SOCP [93] in the minimax probability machine (MPM) [74][135]. This section reviews some basic concepts for supervised learning based on convex optimization, such as SVM, MPM, Fisher discriminant analysis (FDA) [37][30] [69], and DML.

Now, we introduce LP, QP, QCQP, SOCP, and SDP, which have been widely used to model learning problems.

LP is defined by

$$\begin{bmatrix} \min\limits_{\vec{w}} & \vec{c}^T \vec{w} \\ \text{s.t.} & \begin{matrix} G\vec{w} \le \vec{h} \\ A\vec{w} = \vec{b} \end{matrix} \end{bmatrix}, \tag{4.5}$$

where $G \in R^{m \times n}$ and $A \in R^{p \times n}$. That is the convex optimization problem reduces to LP when the objective and constraint functions in the convex optimization problem defined in (4.3) are all affine. The geometric interpretation of the optimal solution for LP is given in Figure 4.3.

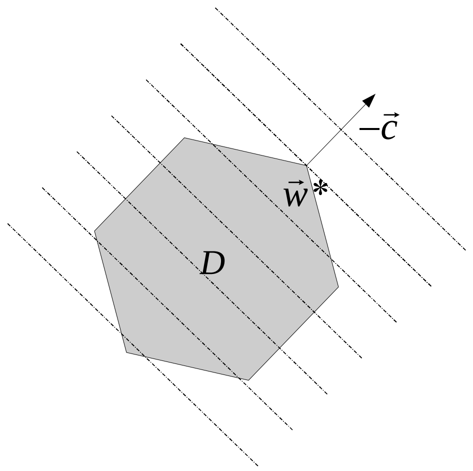


Figure 4.3. The geometric interpretation of the optimal solution $\vec{w}*$ in D for LP defined in (4.5).

QP is defined by

$$\begin{bmatrix} \min\limits_{\vec{w}} & \frac{1}{2}\vec{w}^T P\vec{w} + \vec{q}^T\vec{w} + r \\ \text{s.t.} & \begin{array}{l} G\vec{w} \le \vec{h} \\ A\vec{w} = \vec{b} \end{array} \end{bmatrix}, \tag{4.6}$$

where $P \in S_+^n$, $G \in R^{m\times n}$ and $A \in R^{p\times n}$. Therefore, the convex optimization problem reduces to QP when the objective function in (4.3) is convex quadratic and the constraint functions in (4.3) are all affine. The geometric interpretation of the optimal solution for QP is given in Figure 4.4.

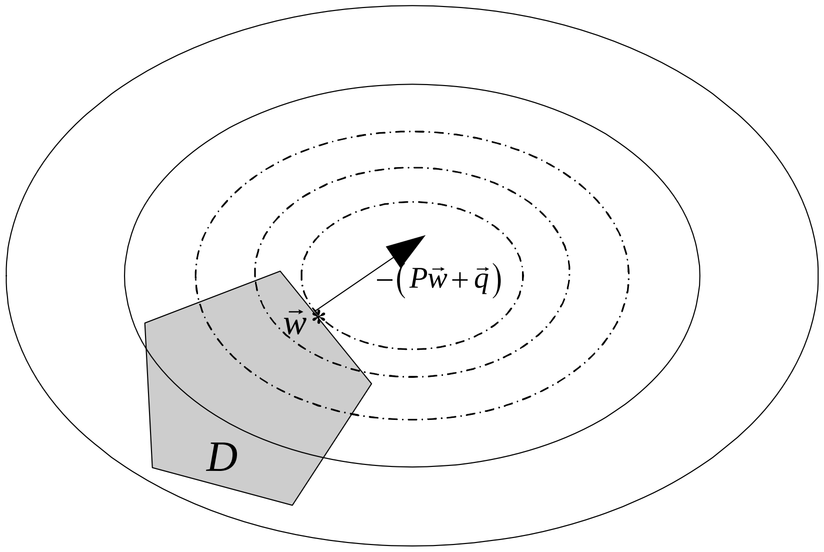


Figure 4.4. The geometric interpretation of the optimal solution $\vec{w}*$ in D for QP defined in (4.6).

If the inequality constraints are not affine but quadratic, (4.6) transfroms to QCQP, i.e.,

$$\begin{bmatrix} \min\limits_{\vec{w}} & \frac{1}{2}\vec{w}^T P_0\vec{w} + \vec{q}_0^T\vec{w} + r_0 \\ \text{s.t.} & \begin{array}{l} \frac{1}{2}\vec{w}^T P_i\vec{w} + \vec{q}_i^T\vec{w} + r_i \le 0, \quad 1 \le i \le m \\ A\vec{w} = \vec{b} \end{array} \end{bmatrix}, \tag{4.7}$$

where $P_i \in S_+^n$ for $0 \le i \le m$.

SOCP has the form

$$\begin{bmatrix} \min\limits_{\vec{w}} & \vec{f}^T\vec{w} \\ \text{s.t.} & \|A_i\vec{w}+b_i\|_{Fro} \le \vec{c}_i^T\vec{w}+d_i, \quad 1\le i\le m \\ & F\vec{w}=\vec{g} \end{bmatrix}, \tag{4.8}$$

where $A_i \in R^{n_i\times n}$, $F \in R^{p\times n}$, $\vec{c}_i \in R^n$, $\vec{g}\in R^p$, $b_i \in R^{n_i}$, and $d_i \in R$. The constraint with the form $\|A\vec{w}+b\| \le \vec{c}^T\vec{w}+d$ is called the second order cone constraint. When $\vec{c}_i = 0$ for all $1\le i\le m$, SOCP transforms to QCQP.

Recently, SDP has become an increasingly important technique in machine learning and many SDP based learning machines have been developed. SDP minimizes a linear function subject to a matrix semidefinite constraint

$$\begin{bmatrix} \min\limits_{\vec{w}} & \vec{c}^T\vec{w} \\ \text{s.t.} & F(\vec{w}) \triangleq F_0+\sum_{i=1}^{n} w_iF_i \ge 0 \end{bmatrix}, \tag{4.9}$$

where $F_i \in S^m$ for all $0\le i\le n$ and $\vec{c}\in R^n$.

Here, we provide a general formula for convex optimization based learning as

$$\begin{bmatrix} \min\limits_{\vec{w},b,\vec{\xi}} & f\left(\vec{w},b,\vec{\xi}\right) \\ \text{s.t.} & y_ic_i\left(\vec{w}^T\vec{x}_i+b\right)\ge\xi_i, \quad 1\le i\le N \end{bmatrix}, \tag{4.10}$$

where $f: R^{L+N+1}\to R$ is a criterion (convex function) for classification; $c_i: R\to R$ for all $1\le i\le N$ are convex constraint functions; $\vec{x}_i\in R^L$ ($1\le i\le N$) are training samples and their class labels are given by $y_i\in\{+1,-1\}$; $\vec{\xi}=[\xi_1,\xi_2,\cdots,\xi_N]^T\in R^N$ are slack variables; and $\vec{w}\in R^L$ and $b\in R$ determine a classification hyperplane, i.e., $\vec{w}^T\vec{x}+b=0$. By defining different classification criteria f and convex constraint functions $c_i|_{i=1}^N$, we can obtain a large number of learning machines, such as SVM, MPM, FDA, and DML. We detail this in the next Section.

STL extends vector based learning algorithms to accept general tensors as input. In STL, we have N training samples $\mathbf{X}_i \in R^{L_1 \times L_2 \times \cdots L_M}$ represented by tensors associated with class label information $y_i \in \{+1, -1\}$. We want to separate positive samples ($y_i = +1$) from negative samples ($y_i = -1$) based on a criterion. This extension is obtained by replacing $\vec{x}_i \in R^L$ ($1 \le i \le N$) and $\vec{w} \in R^L$ with $\mathbf{X}_i \in R^{L_1 \times L_2 \times \cdots L_M}$ ($1 \le i \le N$) and $\vec{w}_k \in R^{L_k}$ ($1 \le k \le M$) in (4.10). Therefore, STL is defined by

$$\left[\begin{array}{ll} \min\limits_{\vec{w}_k|_{k=1}^M, b, \vec{\xi}} & f\left(\vec{w}_k|_{k=1}^M, b, \vec{\xi}\right) \\ \text{s.t.} & y_i c_i \left(\mathbf{X}_i \prod_{k=1}^{M} \times_k \vec{w}_k + b\right) \ge \xi_i, \quad 1 \le i \le N \end{array}\right]. \tag{4.11}$$

There are two main differences between vector based learning and tensor based learning: 1) training samples are represented by vectors in vector based learning, while they are represented by tensors in tensor based learning; and 2) the classification decision function is defined by $\vec{w} \in R^L$ and $b \in R$ in vector based learning ($y(\vec{x}) = \text{sign}\left[\vec{w}^T \vec{x} + b\right]$), while it is defined by $\vec{w}_k \in R^{L_k}$ ($1 \le k \le M$) and $b \in R$ in tensor based learning, i.e., $y(\mathbf{X}) = \text{sign}\left[\mathbf{X} \prod_{k=1}^{M} \times_k \vec{w}_k + b\right]$. In vector based learning, we have a classification hyperplane as $\vec{w}^T \vec{x} + b = 0$. While in tensor based learning, we define a classification hyperplane as $\mathbf{X} \prod_{k=1}^{M} \times_k \vec{w}_k + b = 0$.

The Lagrangian for STL defined in (4.11) is

$$\begin{aligned} L\left(\vec{w}_k|_{k=1}^M, b, \vec{\xi}, \vec{\alpha}\right) &= f\left(\vec{w}_k|_{k=1}^M, b, \vec{\xi}\right) - \sum_{i=1}^{N} \alpha_i \left(y_i c_i \left(\mathbf{X}_i \prod_{k=1}^{M} \times_k \vec{w}_k + b\right) - \xi_i \right) \\ &= f\left(\vec{w}_k|_{k=1}^M, b, \vec{\xi}\right) - \sum_{i=1}^{N} \alpha_i y_i c_i \left(\mathbf{X}_i \prod_{k=1}^{M} \times_k \vec{w}_k + b\right) + \vec{\alpha}^T \vec{\xi} \end{aligned}, \tag{4.12}$$

with Lagrange multipliers $\vec{\alpha} = \left[\alpha_1, \alpha_2, \cdots, \alpha_N\right]^T \ge 0$. The solution is determined by the saddle point of the Lagrangian

$$\max_{\vec{\alpha}} \min_{\vec{w}_k|_{k=1}^M, b, \vec{\xi}} L\left(\vec{w}_k|_{k=1}^M, b, \vec{\xi}, \vec{\alpha}\right), \tag{4.13}$$

The derivative of $L\left(\vec{w}_k\,|_{k=1}^{M}, b, \vec{\xi}, \vec{\alpha}\right)$ with respect to $\vec{w}_j$ is

$$\begin{aligned}
\partial_{\vec{w}_j} L &= \partial_{\vec{w}_j} f\left(\vec{w}_k\,|_{k=1}^{M}, b, \vec{\xi}\right) - \sum_{i=1}^{N} \alpha_i y_i \partial_{\vec{w}_j} c_i\left(\mathbf{X}_i \prod_{k=1}^{M} {}_{\times_k}\, \vec{w}_k + b\right) \\
&= \partial_{\vec{w}_j} f\left(\vec{w}_k\,|_{k=1}^{M}, b, \vec{\xi}\right) - \sum_{i=1}^{N} \alpha_i y_i \frac{dc_i}{dz} \partial_{\vec{w}_j}\left(\mathbf{X}_i \prod_{k=1}^{M} {}_{\times_k}\, \vec{w}_k + b\right) \\
&= \partial_{\vec{w}_j} f\left(\vec{w}_k\,|_{k=1}^{M}, b, \vec{\xi}\right) - \sum_{i=1}^{N} \alpha_i y_i \frac{dc_i}{dz}\left(\mathbf{X}_i \bar{\times}_j\, \vec{w}_j\right),
\end{aligned} \tag{4.14}$$

where $z = \mathbf{X}_i \prod_{k=1}^{M} {}_{\times_k}\, \vec{w}_k + b$.

The derivative of $L\left(\vec{w}_k\,|_{k=1}^{M}, b, \vec{\xi}, \vec{\alpha}\right)$ with respect to b is

$$\begin{aligned}
\partial_b L &= \partial_b f\left(\vec{w}_k\,|_{k=1}^{M}, b, \vec{\xi}\right) - \sum_{i=1}^{N} \alpha_i y_i \partial_b c_i\left(\mathbf{X}_i \prod_{k=1}^{M} {}_{\times_k}\, \vec{w}_k + b\right) \\
&= \partial_b f\left(\vec{w}_k\,|_{k=1}^{M}, b, \vec{\xi}\right) - \sum_{i=1}^{N} \alpha_i y_i \frac{dc_i}{dz} \partial_b\left(\mathbf{X}_i \prod_{k=1}^{M} {}_{\times_k}\, \vec{w}_k + b\right) \\
&= \partial_b f\left(\vec{w}_k\,|_{k=1}^{M}, b, \vec{\xi}\right) - \sum_{i=1}^{N} \alpha_i y_i \frac{dc_i}{dz},
\end{aligned} \tag{4.15}$$

where $z = \mathbf{X}_i \prod_{k=1}^{M} {}_{\times_k}\, \vec{w}_k + b$.

To obtain a solution to STL, we need to set $\partial_{\vec{w}_j} L = 0$ and $\partial_b L = 0$. Accoridng to (4.14), we have

$$\partial_{\vec{w}_j} L = 0 \Rightarrow \partial_{\vec{w}_j} f = \sum_{i=1}^{N} \alpha_i y_i \frac{dc_i}{dz}\left(\mathbf{X}_i \bar{\times}_j\, \vec{w}_j\right). \tag{4.16}$$

According to (4.15), we have

$$\partial_b L = 0 \Rightarrow \partial_b f = \sum_{i=1}^{N} \alpha_i y_i \frac{dc_i}{dz}. \tag{4.17}$$

Based on (4.16), we find the solution to $\vec{w}_j$ depends on $\vec{w}_k$ ($1 \le k \le M$, $k \ne j$). That is we cannot obtain the solution to STL directly. The alternating projection provides a clue to have a solution to STL. The key idea in the alternating projection based optimization for STL is to obtain $\vec{w}_j$ with the given $\vec{w}_k$ ($1 \le k \le M$, $k \ne j$) in an iterative way. The algorithm is given in Table 4.1. The convergence issue is proved in Theorem 4.1.

Table 4.1. Alternating Projection for the Supervised Tensor Learning

Input: Training samples $\mathbf{X}_i |_{i=1}^N \in R^{L_1 \times L_2 \times \ldots \times L_M}$ and their associated class labels $y_i = \{+1, -1\}$.

Output: The parameters $\vec{w}_k |_{k=1}^M \in R^{L_k}$ and $b \in R$, such that the STL objective function $f\left(\vec{w}_k |_{k=1}^M, b, \vec{\xi}\right)$ defined in (4.11) is minimized.

```
1. Set $\vec{w}_k |_{k=1}^M$ be equal to random unit vectors in $R^{L_k}$.
2.     Carry out steps 3–5 iteratively until convergence.
3.     For $j = 1$ to $M$
               Obtain $\vec{w}_j \in R^{L_j}$ by solving
4.             $\begin{bmatrix} \min_{\vec{w}_j, b, \vec{\xi}} & f\left(\vec{w}_j, b, \vec{\xi}\right) \\ \text{s.t.} & y_i c_i \left[ \vec{w}_j^T \left( \mathbf{X}_i \bar{\times}_j \vec{w}_j \right) + b \right] \geq \xi_i, \quad 1 \leq i \leq N \end{bmatrix}$
5.     End
6.     Convergence checking: if $\sum_{k=1}^M \left[ \left| \vec{w}_{k,t}^T \vec{w}_{k,t-1} \right| \left( \left\| \vec{w}_{k,t} \right\|_{Fro}^{-2} \right) - 1 \right] \leq \varepsilon$ ( $\varepsilon = 10^{-6}$ ),
       the calculated $\vec{w}_k |_{k=1}^M$ have converged. Here $\vec{w}_{k,t}$ is the current
       projection vector and $\vec{w}_{k,t-1}$ is the previous projection vector.
7. End
```

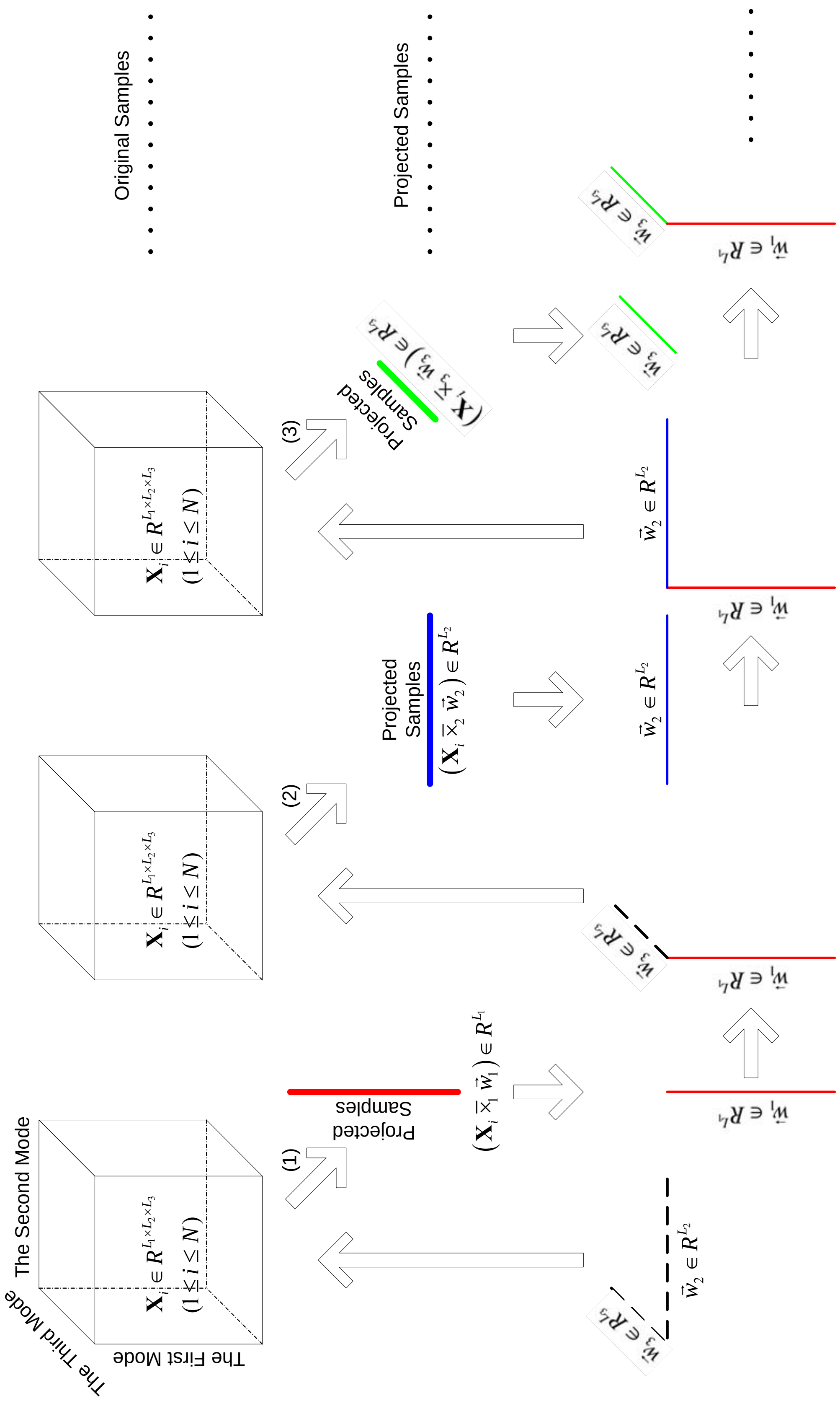
The alternating projection procedure to obtain a solution in STL is illustrated in Table 4.1 and Figure 4.5. In this figure, training samples are represented by third order tensors. The following three steps are conducted iteratively to obtain the solution for STL:

1) Generate the second projection vector $\vec{w}_2$ and third projection vectors $\vec{w}_3$ randomly according to the Step 1 in Table 4.1; project the original training samples (third order tensors) $\mathbf{X}_i \in R^{L_1 \times L_2 \times L_3}$ ($1 \leq i \leq N$) through $\vec{w}_2$ and $\vec{w}_3$ as $\left(\mathbf{X}_i \bar{\times}_1 \vec{w}_1 \right) \in R^{L_1}$; and calculate the first projection vector $\vec{w}_1$ according to the Step 4 in Table 4.1 based on the projected training samples $\left(\mathbf{X}_i \bar{\times}_1 \vec{w}_1 \right)$;

2) Project the original training samples $\mathbf{X}_i |_{i=1}^N$ to the calculated first projection vector $\vec{w}_1$ and the original third projection vector $\vec{w}_3$; and calculate the second

projection vector $\vec{w}_2$ according to the Step 4 in Table 4.1 based on the projected training samples $\left(\mathbf{X}_i \bar{\times}_2 \vec{w}_2\right)$;

3) Project the original training samples $\mathbf{X}_i \mid_{i=1}^{N}$ by the previous $\vec{w}_1$ and $\vec{w}_2$; and calculate $\vec{w}_3$ through the Step 4 in Table 4.1 based on the projected training samples $\left(\mathbf{X}_i \bar{\times}_3 \vec{w}_3\right)$.



The Second Mode
The Third Mode
The First Mode
$\mathbf{X}_i \in R^{L_1 \times L_2 \times L_3}$
$(1 \le i \le N)$
(1)
(2)
(3)
Projected Samples
$(\mathbf{X}_i \bar{\times}_1 \vec{w}_1) \in R^{L_1}$
$(\mathbf{X}_i \bar{\times}_2 \vec{w}_2) \in R^{L_2}$
$(\mathbf{X}_i \bar{\times}_3 \vec{w}_3) \in R^{L_3}$
$\vec{w}_1 \in R^{L_1}$
$\vec{w}_2 \in R^{L_2}$
$\vec{w}_3 \in R^{L_3}$
Original Samples
Projected Samples

Figure 4.5. The third order tensor example for the alternating projection in STL.

Theorem 4.1 The alternating projection based optimization procedure for STL converges.

Proof.

The alternating projection method never increases the function value $f\left(\vec{w}_k\,|_{k=1}^{M}, b, \vec{\xi}\right)$ of STL between two successive iterations, because it can be interpreted as a type of a monotonic algorithm. We can define a continuous function

$$f : \bar{u}_1 \times \bar{u}_2 \times \cdots \times \bar{u}_M \times R \times R^N = \mathop{\times}_{k=1}^{M} \bar{u}_k \times R \times R^N \to R,$$

where $\vec{w}_d \in \bar{u}_d$ and $\bar{u}_d$ is the set, which includes all possible $\vec{w}_d$. The bias $b \in R$ and the slack variables $\vec{\xi} \in R^N$.

With the definition, f has M different mappings:

$$\begin{aligned} g\left(\vec{w}_d^*, b_d^*, \vec{\xi}_d^*\right) &\triangleq \arg\min_{\vec{u}_d \in \bar{u}_d, b, \vec{\xi}} f\left(\vec{w}_d\,|_{d=1}^{M}, b, \vec{\xi}\right) \\ &= \arg\min_{\vec{u}_d \in \bar{u}_d, b, \vec{\xi}} f\left(\vec{w}_d, b, \vec{\xi}; \vec{w}_l\,|_{l=1}^{d-1}, \vec{w}_l\,|_{l=d+1}^{M}\right), \end{aligned}$$

The mapping can be calculated with the given $\vec{w}_l\,|_{l=1}^{d-1}$ in the t^{th} iteration and $\vec{w}_l\,|_{l=d+1}^{M}$ in the $(t-1)^{\text{th}}$ iteration of the for–loop in Step 4 in Table 4.1.

Given an initial $\vec{w}_d \in \bar{u}_d$ ($1 \le d \le M$), the alternating projection generates a sequence of items $\left\{\vec{w}_{d,t}^*, b_{d,t}^*, \vec{\xi}_{d,t}^*; 1 \le d \le M\right\}$ via

$$g\left(\vec{w}_{d,t}^*, b_{d,t}^*, \vec{\xi}_{d,t}^*\right) = \arg\min_{\vec{u}_d \in \bar{u}_d, b, \vec{\xi}} f\left(\vec{w}_d, b, \vec{\xi}; \vec{w}_{l,t}\,|_{l=1}^{d-1}, \vec{w}_{l,t-1}\,|_{l=d+1}^{M}\right),$$

with each $d \in \{1, 2, \cdots M\}$. The sequence has the following relationship:

$$\begin{aligned} a = f(\vec{w}_{1,1}^*, b_{1,1}^*, \vec{\xi}_{1,1}^*) &\ge f(\vec{w}_{2,1}^*, b_{2,1}^*, \vec{\xi}_{2,1}^*) \ge \cdots \ge f(\vec{w}_{M,1}^*, b_{M,1}^*, \vec{\xi}_{M,1}^*) \ge f(\vec{w}_{1,2}^*, b_{1,2}^*, \vec{\xi}_{1,2}^*) \ge \\ \cdots \ge f(\vec{w}_{1,t}^*, b_{1,t}^*, \vec{\xi}_{1,t}^*) &\ge f(\vec{w}_{2,t}^*, b_{2,t}^*, \vec{\xi}_{2,t}^*) \ge \cdots \ge f(\vec{w}_{1,T}^*, b_{1,T}^*, \vec{\xi}_{1,T}^*) \ge f(\vec{w}_{2,T}^*, b_{2,T}^*, \vec{\xi}_{2,T}^*) \ge \\ &\cdots \ge f(\vec{w}_{M,T}^*, b_{M,T}^*, \vec{\xi}_{M,T}^*) = b. \end{aligned}$$

where $T \to +\infty$. Here both a and b are limited values in the R space.

The alternating projection in STL can be illustrated by a composition of M sub–algorithms, defined as

$$\Omega_d : \left(\vec{w}_d\,|_{d=1}^{M}, b, \vec{\xi}\right) \mapsto \prod_{l=1}^{d-1} {}_{\times_l}\, \vec{w}_l \times \text{Map}\left(\vec{w}_d, b, \vec{\xi}\right) \prod_{l=d+1}^{M} {}_{\times_l}\, \vec{w}_l\,.$$

It follows that $\Omega \triangleq \Omega_1 \circ \Omega_2 \circ \cdots \circ \Omega_M = \overset{M}{\underset{d=1}{\circ}} \Omega_d$ is a closed algorithm whenever all $\overline{u}_d$ are compact. All sub–algorithms $g\left(\vec{w}_d^*, b_d^*, \vec{\xi}_d^*\right)$ decrease the value of f, so it should be clear that Ω is monotonic with respect to f.

Consequently, we can say that the alternating projection method to optimize STL defined in (4.11) converges. ■

Supervised Tensor Learning: Examples

Based on the proposed STL and its alternating projection training algorithm, a large number of tensor based learning algorithms can be obtained by combining STL with different learning criteria, such as SVM, MPM, DML, and FDA.

Support Vector Machine vs. Support Tensor Machine

SVM [161][15][130][128][137][138][41] finds a classification hyperplane, which maximizes the margin between positive samples and negative samples, as shown in Figure 4.6.

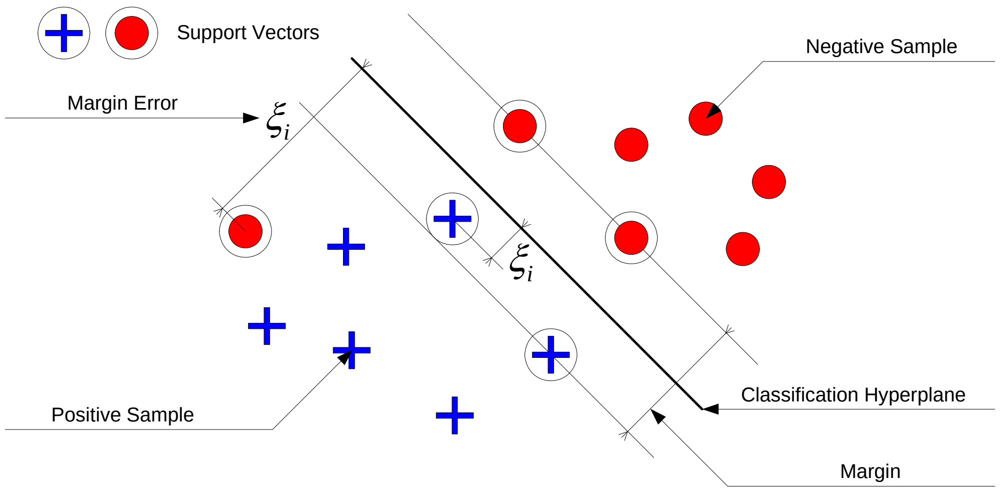


Figure 4.6. SVM maximizes the margin between positive and negative training samples.

Suppose there are N training samples $\vec{x}_i \in R^L$ $(1 \le i \le N)$ associated with class lables $y_i \in \{+1, -1\}$. The traditional SVM [161][15], i.e., soft margin SVM, finds a projection vector $\vec{w} \in R^L$ and a bias $b \in R$ through

$$\left[\begin{array}{ll} \min\limits_{\vec{w},b,\vec{\xi}} & J_{C-SVM}\left(\vec{w},b,\vec{\xi}\right) = \frac{1}{2}\|\vec{w}\|_{Fro}^2 + c\sum_{i=1}^{N}\xi_i \\ \text{s.t.} & y_i\left[\vec{w}^T\vec{x}_i + b\right] \ge 1 - \xi_i, \quad 1 \le i \le N \\ & \vec{\xi} \ge 0 \end{array} \right] \tag{4.18}$$

where $\vec{\xi} = \left[\xi_1, \xi_2, \cdots, \xi_N\right]^T \in R^N$ is the vector of all slack variables to deal with the linearly non–separable problem. The ξ_i ($1 \le i \le N$) is also called the marginal error for the ith training sample, as shown in Figure 4.6. The margin is $2/\|\vec{w}\|_{Fro}$. When the classification problem is linearly separable, we can set $\vec{\xi} = 0$. The decision function for classification is $y\left(\vec{x}\right) = \operatorname{sign}\left[\vec{w}^T \vec{x} + b\right]$.

The Lagrangian of (4.18) is

$$\begin{aligned} L\left(\vec{w}, b, \vec{\xi}, \vec{\alpha}, \vec{\kappa}\right) &= \frac{1}{2}\|\vec{w}\|_{Fro}^2 + c\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\left(y_i\left[\vec{w}^T\vec{x}_i + b\right] - 1 + \xi_i\right) - \sum_{i=1}^{N}\kappa_i\xi_i \\ &= \frac{1}{2}\vec{w}^T\vec{w} + c\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i y_i \vec{w}^T\vec{x}_i - b\vec{\alpha}^T\vec{y} + \sum_{i=1}^{N}\alpha_i - \vec{\alpha}^T\vec{\xi} - \vec{\kappa}^T\vec{\xi} \end{aligned} \tag{4.19}$$

with Lagrangian multipliers $\alpha_i \ge 0$, $\kappa_i \ge 0$ for $1 \le i \le N$. The solution is determined by the saddle point of the Lagrangian

$$\max_{\vec{\alpha}, \vec{\kappa}} \min_{\vec{w}, b, \vec{\xi}} L\left(\vec{w}, b, \vec{\xi}, \vec{\alpha}, \vec{\kappa}\right). \tag{4.20}$$

This can be achieved by

$$\begin{aligned} &\partial_{\vec{w}} L = 0 \Rightarrow \vec{w} = \sum_{i=1}^{N}\alpha_i y_i \vec{x}_i \\ &\partial_b L = 0 \Rightarrow \vec{\alpha}^T\vec{y} = 0 \\ &\partial_{\vec{\xi}} L = 0 \Rightarrow c - \vec{\alpha} - \vec{\kappa} = 0. \end{aligned} \tag{4.21}$$

Based on (4.21), we can have the dual problem of (4.18),

$$\begin{bmatrix} \max\limits_{\vec{\alpha}} \;\; J_D\left(\vec{\alpha}\right) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \vec{x}_i^T \vec{x}_j \alpha_i \alpha_j + \sum_{i=1}^{N}\alpha_i \\ \text{s.t.} \;\; \begin{array}{l} \vec{\alpha}^T\vec{y} = 0 \\ 0 \le \vec{\alpha} \le c \end{array} \end{bmatrix}. \tag{4.22}$$

Set $P = \left[y_i y_j \vec{x}_i^T \vec{x}_j\right]_{1 \le i, j \le N}$, $\vec{q} = \vec{1}_{N\times 1}$, $A = \vec{y}$, $\vec{b} = 0$, $G = \left[I_{N\times N}, -I_{N\times N}\right]^T$, and $\vec{h} = \left[c\vec{1}_{N\times 1}^T, \vec{0}_{N\times 1}^T\right]^T$ in (4.6), we can see that the dual problem of (4.18) in SVM is a QP.

In the soft margin SVM defined in (4.18), the constant c determines the tradeoff between 1) maximizing the margin between positive and negative samples and 2) minimizing the training error. The constant c is not intuitive. Therefore, Bcholkopf et al. [130][128] developed the nu–SVM by replacing the unintuitive parameter c with an intuitive parameter ν as

$$\left[\begin{array}{ll}\min\limits_{\vec{w},b,\vec{\xi},\rho} & J_{\nu-SVM}\left(\vec{w},b,\vec{\xi},\rho\right)=\frac{1}{2}\|\vec{w}\|_{Fro}^{2}+\frac{1}{N}\sum_{i=1}^{N}\xi_i-\nu\rho \\ & y_i\left[\vec{w}^T\vec{x}_i+b\right]\geq\rho-\xi_i,\quad 1\leq i\leq N \\ \text{s.t.} & \vec{\xi}\geq 0, \\ & \rho\geq 0\end{array}\right]. \tag{4.23}$$

The significance of ν in nu–SVM defined in (4.23) is that it controls the number of support vectors and the marginal errors.

Suykens and Vandewalle [137][138] simplified the soft margin SVM as the least squares SVM,

$$\left[\begin{array}{l}\min\limits_{\vec{w},b,\vec{\varepsilon}}\ J_{LS-SVM}\left(\vec{w},b,\vec{\varepsilon}\right)=\frac{1}{2}\|\vec{w}\|_{Fro}^{2}+\frac{\gamma}{2}\vec{\varepsilon}^T\vec{\varepsilon} \\ \text{s.t.}\ \ y_i\left[\vec{w}^T\vec{x}_i+b\right]=1-\varepsilon_i,\quad 1\leq i\leq N\end{array}\right]. \tag{4.24}$$

Here the penalty $\gamma>0$. There are two differences between the soft margin SVM defined in (4.18) and the least squares SVM defined in (4.24): 1) inequality constraints are replaced by equalities; and 2) the loss $\sum_{i=1}^{N}\xi_i$ ($\xi_i\geq 0$) is replaced by square loss. These two modifications make the solution of the least square SVM be more easily obtained compared with soft margin SVM.

According to statistical learning theory, a learning machine performs well when the number of training samples is larger than the complexity of the model. Moreover, the model's complexity and the number of parameters to describe the model are always in direct proportion. In computer vision research, objects are usually represented by general tensors as described in Chapter 3 and the number of training samples is limited. Therefore, it is reasonable to have the tensor extension of SVM i.e., the support tensor machine (STM), which uses fewer parameters than SVM. Based on (4.18) and STL defined in (4.11), it is not difficult to obtain the tensor extension of the soft margin SVM, i.e., the soft margin STM.

Suppose we have training samples $\mathbf{X}_i\in R^{L_1\times L_2\times\cdots L_M}$ ($1\leq i\leq N$) and their corresponding class labels $y_i\in\{+1,-1\}$. The decision function is defined by $y(\mathbf{X})=\text{sign}\left[\mathbf{X}\prod_{k=1}^{M}\times_k\vec{w}_k+b\right]$, where the projection vectors $\vec{w}_k\in R^{L_k}$ ($1\leq k\leq M$) and the bias b in soft margin STM are obtained from

$$\left[\begin{array}{ll} \min\limits_{\vec{w}_k|_{k=1}^M, b, \vec{\xi}} & J_{C-STM}\left(\vec{w}_k|_{k=1}^M, b, \vec{\xi}\right) = \frac{1}{2}\left\|\prod_{k=1}^{M} \otimes \vec{w}_k\right\|_{Fro}^2 + c\sum_{i=1}^{N}\xi_i \\ \text{s.t.} & y_i\left[\mathbf{X}_i \prod_{k=1}^{M} \times_k \vec{w}_k + b\right] \geq 1-\xi_i, \quad 1 \leq i \leq N \\ & \vec{\xi} \geq 0 \end{array}\right]. \tag{4.25}$$

Here, $\vec{\xi} = \left[\xi_1, \xi_2, \cdots, \xi_N\right]^T \in R^N$ is the vector of all slack variables to deal with the linearly non–separable problem.

The Lagrangian for this problem is

$$\begin{aligned} L\left(\vec{w}_k|_{k=1}^M, b, \vec{\xi}, \vec{\alpha}, \vec{\kappa}\right) &= \frac{1}{2}\left\|\prod_{k=1}^{M} \otimes \vec{w}_k\right\|_{Fro}^2 + c\sum_{i=1}^{N}\xi_i \\ &\quad - \sum_{i=1}^{N}\alpha_i\left(y_i\left[\mathbf{X}_i \prod_{k=1}^{M} \times_k \vec{w}_k + b\right] - 1 + \xi_i\right) - \sum_{i=1}^{N}\kappa_i\xi_i \\ &= \frac{1}{2}\prod_{k=1}^{M}\vec{w}_k^T\vec{w}_k + c\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i y_i\left(\mathbf{X}_i\prod_{k=1}^{M}\times_k \vec{w}_k\right) \\ &\quad - b\vec{\alpha}^T\vec{y} + \sum_{i=1}^{N}\alpha_i - \vec{\alpha}^T\vec{\xi} - \vec{\kappa}^T\vec{\xi} \end{aligned} \tag{4.26}$$

with Lagrangian multipliers $\alpha_i \geq 0$, $\kappa_i \geq 0$ for $1 \leq i \leq N$. The solution is determined by the saddle point of the Lagrangian

$$\max_{\vec{\alpha}, \vec{\kappa}} \min_{\vec{w}_k|_{k=1}^M, b, \vec{\xi}} L\left(\vec{w}_k|_{k=1}^M, b, \vec{\xi}, \vec{\alpha}, \vec{\kappa}\right). \tag{4.27}$$

This can be achieved by

$$\begin{aligned} &\partial_{\vec{w}_j} L = 0 \Rightarrow \vec{w}_j = \frac{1}{\prod\limits_{k=1}^{k \neq j}\vec{w}_k^T\vec{w}_k}\sum_{i=1}^{N}\alpha_i y_i\left(\mathbf{X}_i \bar{\times}_j \vec{w}_j\right) \\ &\partial_b L = 0 \Rightarrow \vec{\alpha}^T\vec{y} = 0 \\ &\partial_{\vec{\xi}} L = 0 \Rightarrow c - \vec{\alpha} - \vec{\kappa} = 0 \end{aligned} . \tag{4.28}$$

The first equation in (4.28) shows that the solution of $\vec{w}_j$ depends on $\vec{w}_k$ ($1 \leq k \leq M$, $k \neq j$). That is we cannot obtain the solution for the soft margin STM directly. This point has been pointed out in the STL framework developed in §4.2. Therefore, we use the proposed alternating projection method in STL to obtain the solution of the soft margin STM. To have the alternating projection method for the soft margin STM, we need to replace the Step 4 in Table 4.1 by the following optimization problem,

$$\left[\begin{array}{ll}\min\limits_{\vec{w}_j,b,\vec{\xi}} & J_{C-STM}\left(\vec{w}_j,b,\vec{\xi}\right)=\frac{\eta}{2}\left\|\vec{w}_j\right\|_{Fro}^2+c\sum_{i=1}^{N}\xi_i \\ \text{s.t.} & \begin{array}{l} y_i\left[\vec{w}_j^T\left(\mathbf{X}_i\times_j\vec{w}_j\right)+b\right]\ge 1-\xi_i,\quad 1\le i\le N \\ \vec{\xi}\ge 0\end{array}\end{array}\right], \tag{4.29}$$

where $\eta=\prod_{1\le k\le M}^{k\ne j}\left\|\vec{w}_k\right\|_{Fro}^2$.

The problem defined in (4.29) is the standard soft margin SVM defined in (4.18). Based on nu–SVM defined in (4.23) and STL defined in (4.11), we can also have the tensor extension of the nu–SVM, i.e., nu–STM,

$$\left[\begin{array}{ll}\min\limits_{\vec{w}_k|_{k=1}^M,b,\vec{\xi},\rho} & J_{\nu-STM}\left(\vec{w}_k|_{k=1}^M,b,\vec{\xi},\rho\right)=\frac{1}{2}\left\|\prod_{k=1}^{M}{}_{\otimes}\vec{w}_k\right\|_{Fro}^2+\frac{1}{N}\sum_{i=1}^{N}\xi_i-\nu\rho \\ \text{s.t.} & \begin{array}{l} y_i\left[\mathbf{X}_i\prod_{k=1}^{M}{}_{\times_k}\vec{w}_k+b\right]\ge\rho-\xi_i,\quad 1\le i\le N \\ \vec{\xi}\ge 0, \\ \rho\ge 0\end{array}\end{array}\right]. \tag{4.30}$$

Here, $\nu\ge 0$ is a constant. The Lagrangian for this problem is

$$\begin{aligned}L\left(\vec{w}_k|_{k=1}^M,b,\vec{\xi},\rho,\vec{\alpha},\vec{\kappa},\tau\right)&=\frac{1}{2}\left\|\prod_{k=1}^{M}{}_{\otimes}\vec{w}_k\right\|_{Fro}^2+\frac{1}{N}\sum_{i=1}^{N}\xi_i-\nu\rho-\tau\rho\\&\quad-\sum_{i=1}^{N}\alpha_i\left(y_i\left[\mathbf{X}_i\prod_{k=1}^{M}{}_{\times_k}\vec{w}_k+b\right]-\rho+\xi_i\right)-\vec{\kappa}^T\vec{\xi}\\&=\frac{1}{2}\prod_{k=1}^{M}\vec{w}_k^T\vec{w}_k+\frac{1}{N}\sum_{i=1}^{N}\xi_i-\sum_{i=1}^{N}\alpha_i y_i\left(\mathbf{X}_i\prod_{k=1}^{M}{}_{\times_k}\vec{w}_k\right)\\&\quad-b\vec{\alpha}^T\vec{y}+\rho\sum_{i=1}^{N}\alpha_i-\vec{\alpha}^T\vec{\xi}-\vec{\kappa}^T\vec{\xi}-\nu\rho-\tau\rho,\end{aligned} \tag{4.31}$$

with Lagrangian multipliers $\tau\ge 0$ and $\alpha_i\ge 0$, $\kappa_i\ge 0$ for $1\le i\le N$. The solution is determined by the saddle point of the Lagrangian

$$\max_{\vec{\alpha},\vec{\kappa},\tau}\min_{\vec{w}_k|_{k=1}^M,b,\vec{\xi},\rho}L\left(\vec{w}_k|_{k=1}^M,b,\vec{\xi},\rho,\vec{\alpha},\vec{\kappa},\tau\right). \tag{4.32}$$

Similar to the soft margin STM, the solution of $\vec{w}_j$ depends on $\vec{w}_k$ ($1\le k\le M$, $k\ne j$), because

$$\partial_{\vec{w}_j}L=0\Rightarrow\vec{w}_j=\frac{1}{\prod_{k=1}^{k\ne j}\vec{w}_k^T\vec{w}_k}\sum_{i=1}^{N}\alpha_i y_i\left(\mathbf{X}_i\times_j\vec{w}_j\right). \tag{4.33}$$

Therefore, we use the proposed alternating projection method in STL to obtain the solution of nu–STM. To have the alternating projection method for nu–STM, we need to replace the Step 4 in Table 4.1 by the following optimization problem,

$$\left[\begin{array}{ll} \min\limits_{\vec{w}_j, b, \vec{\xi}, \rho} & J_{\nu-STM}\left(\vec{w}_j, b, \vec{\xi}, \rho\right) = \frac{\eta}{2}\left\|\vec{w}_j\right\|_{Fro}^2 + \frac{1}{N}\sum_{i=1}^{N}\xi_i - \nu\rho \\ & y_i\left[\vec{w}_j^T\left(\mathbf{X}_i \times_j \vec{w}_j\right) + b\right] \geq \rho - \xi_i, \quad 1 \leq i \leq N \\ \text{s.t.} & \vec{\xi} \geq 0 \\ & \rho \geq 0 \end{array}\right], \tag{4.34}$$

where $\eta = \prod_{1\le k\le M}^{k\neq j}\left\|\vec{w}_k\right\|_{Fro}^2$.

The problem defined in (4.34) is the standard nu–SVM defined in (4.23).

Based on the least squares SVM defined in (4.24) and STL defined in (4.11), we can also have the tensor extension of the least square SVM, i.e., least square STM,

$$\left[\begin{array}{ll} \min\limits_{\vec{w}_k|_{k=1}^M, b, \vec{\varepsilon}} & J_{LS-STM}\left(\vec{w}_k|_{k=1}^M, b, \vec{\varepsilon}\right) = \frac{1}{2}\left\|\prod_{k=1}^{M}\otimes\vec{w}_k\right\|_{Fro}^2 + \frac{\gamma}{2}\vec{\varepsilon}^T\vec{\varepsilon} \\ \text{s.t.} & y_i\left[\mathbf{X}_i\prod_{k=1}^{M}\times_k\vec{w}_k + b\right] = 1 - \varepsilon_i, \quad 1 \leq i \leq N \end{array}\right], \tag{4.35}$$

where $\gamma > 0$ is a constant. Similar to the soft margin STM and nu–STM, there is no closed form solution for least squares STM. We use the alternating projection method in STL to obtain the solution of the least squares STM. To have the alternating projection method for the least squares STM, we need to replace the Step 4 in Table 4.1 by the following optimization problem,

$$\left[\begin{array}{ll} \min\limits_{\vec{w}_j, b, \vec{\varepsilon}} & J_{LS-STM}\left(\vec{w}_k|_{k=1}^M, b, \vec{\varepsilon}\right) = \frac{\eta}{2}\left\|\vec{w}_j\right\|_{Fro}^2 + \frac{\gamma}{2}\vec{\varepsilon}^T\vec{\varepsilon} \\ \text{s.t.} & y_i\left[\vec{w}_j^T\left(\mathbf{X}_i \times_j \vec{w}_j\right) + b\right] = 1 - \varepsilon_i, \quad 1 \leq i \leq N \end{array}\right], \tag{4.36}$$

where $\eta = \prod_{1\le k\le M}^{k\neq j}\left\|\vec{w}_k\right\|_{Fro}^2$.

Theorem 4.2 In STM, the decision function is $y(\mathbf{X}) = \text{sign}\left[\mathbf{X}\prod_{k=1}^{M}\times_k\vec{w}_k + b\right]$ with $\left\|\prod_{k=1}^{M}\otimes\vec{w}_k\right\|_{Fro}^2 \leq \Lambda^2$ and $\left\|\mathbf{X}\right\|_{Fro}^2 \leq R^2$. Let $\rho > 0$ and ν is the fraction of training samples with a margin smaller than $\rho/|\Lambda|$. When STM is obtained from N training samples $\left\|\mathbf{X}_i\right\|_{Fro}^2 \leq R^2$ ($1 \leq i \leq N$), sampled from a distribution P with probability at least $1-\delta$ $(0<\delta<1)$, the misclassification probability of a test sample sampled from P is bounded by

$$\nu+\sqrt{\frac{\lambda}{N}\left(\frac{R^2\Lambda^2}{\rho^2}\ln^2 N+\ln\frac{1}{\delta}\right)}, \tag{4.37}$$

where λ is a universal constant.

Proof: This is a direct conclusion from the theorem on the margin error bound introduced in [128]. More information about other error bounds in SVM can be found in [3].

Minimax Probability Machine vs. Tensor Minimax Probability Machine

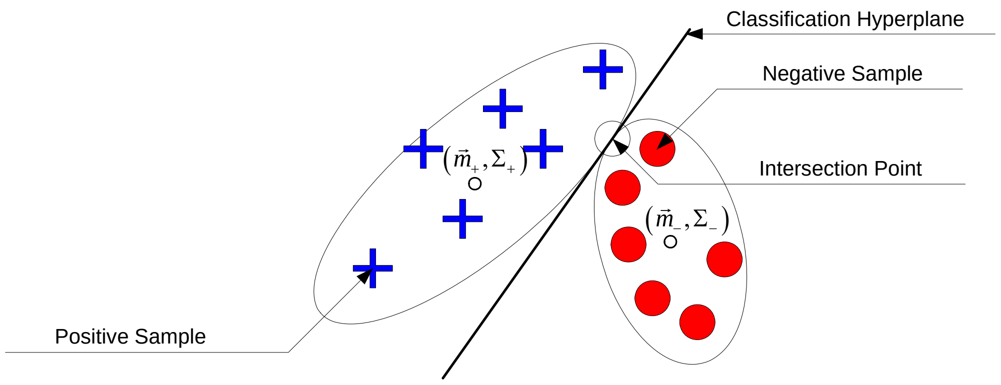


Figure 4.7. MPM separates positive samples from negative samples by maximizing the probability of the correct classification for future samples. The intersection point minimizes the maximum of the Mahalanobis distances between positive and negative samples, i.e., it has the same Mahalanobis distances to the mean of the positive samples and the mean of the negative samples.

The minimax probability machine (MPM) [74][135] has become popular. It is reported to outperform the conventional SVM consistently and therefore has attracted attention as a promising supervised learning algorithm. MPM focuses on finding a decision hyper–plane, which is $H(\vec{w},b)=\{\vec{x}\mid\vec{w}^T\vec{x}+b=0\}$, to separate positive samples from negative samples (a binary classification problem) with maximal probability with respect to all distributions modelled by given means and covaraince, as shown in Figure 4.7. MPM maximizes the probability of the correct classification rate (classification accuracy) on future samples. For Gaussian distributed samples, it minimizes the maximum of the Mahalanobis distances of the positive samples and the negative samples. With given positive samples $\vec{x}_i$ ($y_i = +1$) and negative samples $\vec{x}_i$ ($y_i = -1$), MPM is defined as,

$$\begin{bmatrix} \max\limits_{\vec{w},b,\delta} & J_{MPM}\left(\vec{w},b,\delta\right) = \delta \\ \text{s.t.} & \begin{matrix} \inf\limits_{\vec{x}_i(y_i=+1)\sim(\vec{m}_+,\Sigma_+)} \Pr\left\{\vec{w}^T\vec{x}_i + b \ge 0\right\} \ge \delta \\ \inf\limits_{\vec{x}_i(y_i=-1)\sim(\vec{m}_-,\Sigma_-)} \Pr\left\{\vec{w}^T\vec{x}_i + b \le 0\right\} \ge \delta \end{matrix} \end{bmatrix}. \tag{4.38}$$

Here, the notation $\vec{x}_i\left(y_i = +1\right) \sim \left(\vec{m}_+, \Sigma_+\right)$ means the class distribution of the positive samples has the mean $\vec{m}_+$ and covariance Σ_+, and similarly for the notation $\vec{x}_i\left(y_i = -1\right) \sim \left(\vec{m}_-, \Sigma_-\right)$. The classification decision function is given by $y\left(\vec{x}\right) = \text{sign}\left[\vec{w}^T\vec{x} + b\right]$.

Recently, based on the powerful Marshall and Olkin's theorem [102], Popescu and Bertsimas [120] proved a probability bound,

$$\sup_{\vec{x}\sim(\vec{m},\Sigma)} \Pr\left\{\vec{x} \in S\right\} = \frac{1}{1+d^2} \quad \text{with} \quad d^2 = \inf_{\vec{x}\in S}\left(\vec{x}-\vec{m}\right)^T \Sigma^{-1}\left(\vec{x}-\vec{m}\right), \tag{4.39}$$

where $\vec{x}$ stands for a random vector, S is a given convex set, and the supremum is taken over all distributions for $\vec{x}$ with the mean value as $\vec{m}$ and the covariance matrix Σ. Based on this result, Lanckriet et al. [74] reformulated (4.38) as:

$$\begin{bmatrix} \max\limits_{\vec{w},b,\kappa} & J_{MPM}\left(\vec{w},b,\kappa\right) = \kappa \\ \text{s.t.} & \begin{matrix} \vec{w}^T\vec{m}_+ + b \ge +\kappa\sqrt{\vec{w}^T\Sigma_+\vec{w}}, & y_i = +1 \\ \vec{w}^T\vec{m}_- + b \le -\kappa\sqrt{\vec{w}^T\Sigma_-\vec{w}}, & y_i = -1 \end{matrix} \end{bmatrix}, \tag{4.40}$$

where the constraint functions in (4.40) are second order cone functions. There MPM is an SOCP. This problem can be further simplified as

$$\begin{bmatrix} \min\limits_{\vec{w}} & J_{MPM}\left(\vec{w}\right) = \sqrt{\vec{w}^T\Sigma_+\vec{w}} + \sqrt{\vec{w}^T\Sigma_-\vec{w}} \\ \text{s.t.} & \vec{w}^T\left(\vec{m}_+ - \vec{m}_-\right) = 1 \end{bmatrix}, \tag{4.41}$$

where b is determined by

$$b^* = \left(\vec{w}^*\right)^T \vec{m}_+ - \frac{\sqrt{\left(\vec{w}^*\right)^T \Sigma_+ \left(\vec{w}^*\right)}}{\sqrt{\left(\vec{w}^*\right)^T \Sigma_+ \left(\vec{w}^*\right)} + \sqrt{\left(\vec{w}^*\right)^T \Sigma_- \left(\vec{w}^*\right)}}. \tag{4.42}$$

In computer vision research, many objects are represented by tensors. To match the input requirments in MPM, we need to vectorize the tensors to vectors. When training samples are limited, the vectorization will be a disaster. This is because MPM meets the matrix singular problem seriously (the ranks of Σ_+ and Σ_- are deficient). To reduce this problem, we propose the tensor extension of MPM, i.e., tensor MPM (TMPM). TMPM is a combination of MPM and STL.

Suppose we have training samples $\mathbf{X}_i \in R^{L_1 \times L_2 \times \cdots L_M}$ ($1 \le i \le N$) and their corresponding class labels $y_i \in \{+1, -1\}$. The decision function is given by $y(\mathbf{X}) = \text{sign}\left[\mathbf{X} \prod_{k=1}^{M} \times_k \vec{w}_k + b\right]$. Projection vectors $\vec{w}_k \in R^{L_k}$ ($1 \le k \le M$) and the bias b in TMPM are obtained from

$$\left[\begin{array}{ll} \max\limits_{\vec{w}_k|_{k=1}^{M}, b, \kappa} & J_{MPM}\left(\vec{w}_k|_{k=1}^{M}, b, \kappa\right) = \kappa \\ \text{s.t.} & \begin{array}{l} \frac{1}{N_+}\left(\sum_{i=1}^{N}\left[\mathrm{I}(y_i = +1)\mathbf{X}_i \prod_{k=1}^{M} \times_k \vec{w}_k\right]\right) + b \ge +\kappa \sup\limits_{1 \le l \le M} \sqrt{\vec{w}_l^T \Sigma_{+;l} \vec{w}_l} \\ \frac{1}{N_-}\left(\sum_{i=1}^{N}\left[\mathrm{I}(y_i = -1)\mathbf{X}_i \prod_{k=1}^{M} \times_k \vec{w}_k\right]\right) + b \le -\kappa \sup\limits_{1 \le l \le M} \sqrt{\vec{w}_l^T \Sigma_{-;l} \vec{w}_l} \end{array} \end{array}\right], \tag{4.43}$$

where $\Sigma_{+;l}$ is the covariance matrix of the projected samples $\left(\mathbf{X}_i \bar{\times}_{-l} \vec{w}_l\right)$ for all $y_i = +1$ and $\Sigma_{-;l}$ is the covariance matrix of the projected samples $\left(\mathbf{X}_i \bar{\times}_{-l} \vec{w}_l\right)$ for all $y_i = -1$. The function $\mathrm{I}(y_i = +1)$ is 1 if y_i is $+1$, otherwise 0. The function $\mathrm{I}(y_i = -1)$ is 1 if y_i is -1, otherwise 0. This problem can be simplified as

$$\left[\begin{array}{ll} \max\limits_{\vec{w}_k|_{k=1}^{M}, b, \kappa} & J_{MPM}\left(\vec{w}_k|_{k=1}^{M}, b, \kappa\right) = \kappa \\ \text{s.t.} & \begin{array}{l} \mathbf{M}_+ \prod_{k=1}^{M} \times_k \vec{w}_k + b \ge +\kappa \sup\limits_{1 \le l \le M} \sqrt{\vec{w}_l^T \Sigma_{+;l} \vec{w}_l} \\ \mathbf{M}_- \prod_{k=1}^{M} \times_k \vec{w}_k + b \le -\kappa \sup\limits_{1 \le l \le M} \sqrt{\vec{w}_l^T \Sigma_{-;l} \vec{w}_l} \end{array} \end{array}\right], \tag{4.44}$$

where $\mathbf{M}_+ = \left(1/N_+\right)\sum_{i=1}^{N}\left[\mathrm{I}(y_i = +1)\mathbf{X}_i\right]$, $\mathbf{M}_- = \left(1/N_-\right)\sum_{i=1}^{N}\left[\mathrm{I}(y_i = -1)\mathbf{X}_i\right]$, and N_+ (N_-) is the number of positive (negative) samples.

The Lagrangian for this problem is

$$
\begin{aligned}
L\left(\vec{w}_k\,|_{k=1}^{M}, b, \kappa, \vec{\alpha}\right) &= -\kappa - \alpha_1\left(\mathbf{M}_+ \prod_{k=1}^{M} \times_k \vec{w}_k + b - \kappa \sup_{1\le l\le M} \sqrt{\vec{w}_l^T \Sigma_{+;l} \vec{w}_l}\right) \\
&\quad + \alpha_2\left(\mathbf{M}_- \prod_{k=1}^{M} \times_k \vec{w}_k + b + \kappa \sup_{1\le l\le M} \sqrt{\vec{w}_l^T \Sigma_{-;l} \vec{w}_l}\right) \\
&= -\kappa - \frac{\alpha_1 b}{N_+} + \frac{\alpha_2 b}{N_-} - \alpha_1 \mathbf{M}_+ \prod_{k=1}^{M} \times_k \vec{w}_k + \alpha_2 \mathbf{M}_- \prod_{k=1}^{M} \times_k \vec{w}_k \\
&\quad + \frac{\alpha_1 \kappa}{N_+} \sup_{1\le l\le M} \sqrt{\vec{w}_l^T \Sigma_{+;l} \vec{w}_l} + \frac{\alpha_2 \kappa}{N_-} \sup_{1\le l\le M} \sqrt{\vec{w}_l^T \Sigma_{-;l} \vec{w}_l}
\end{aligned}
\tag{4.45}
$$

with Lagrangian multipliers $\alpha_i \ge 0$ ($i = 1, 2$). The solution is determined by the saddle point of the Lagrangian

$$
\max_{\vec{\alpha}} \min_{\vec{w}_k|_{k=1}^{M}, b, \kappa} L\left(\vec{w}_k\,|_{k=1}^{M}, b, \kappa, \vec{\alpha}\right). \tag{4.46}
$$

This can be achieved by setting $\partial_{\vec{w}_j} L = 0$, $\partial_b L = 0$, and $\partial_\kappa L = 0$. It is not difficult to find that the solution of $\vec{w}_j$ depends on $\vec{w}_k$ ($1 \le k \le M$, $k \ne j$). Therefore, there is no closed form solution for TMPM. We use the proposed alternating projection method in STL to obtain the solution of TMPM. To have the alternating projection method for TMPM, we need to replace the Step 4 in Table 4.1 by the following optimization problem,

$$
\left[
\begin{array}{ll}
\max\limits_{\vec{w}_j, b, \kappa} & J_{MPM}\left(\vec{w}_j, b, \kappa\right) = \kappa \\
\text{s.t.} & \begin{array}{l} \vec{w}_j^T \left(\mathbf{M}_+ \bar{\times}_j \vec{w}_j\right) + b \ge +\kappa \sqrt{\vec{w}_j^T \Sigma_{+;j} \vec{w}_j} \\ \vec{w}_j^T \left(\mathbf{M}_- \bar{\times}_j \vec{w}_j\right) + b \le -\kappa \sqrt{\vec{w}_j^T \Sigma_{-;j} \vec{w}_j} \end{array}
\end{array}
\right]. \tag{4.47}
$$

This problem is the standard MPM defined in (4.40).

Fisher Discriminant Analysis vs. Tensor Fisher Discriminant Analysis

Fisher discriminant analysis (FDA) [37][30][69] has been widely applied for classification. Suppose there are N training samples $\vec{x}_i \in R^L$ ($1 \le i \le N$) associated with their class labels $y_i \in \{+1, -1\}$. There are N_+ positive training samples and their mean is $\vec{m}_+ = (1/N_+)\sum_{i=1}^{N}\left[\mathrm{I}(y_i = +1)\vec{x}_i\right]$; there are N_- negative training samples and their mean can be calculated from $\vec{m}_- = (1/N_-)\sum_{i=1}^{N}\left[\mathrm{I}(y_i = -1)\vec{x}_i\right]$; the mean of all training samples is

$\vec{m} = (1/N)\sum_{i=1}^{N}\vec{x}_i$; and the covariance matrix of all training samples is Σ. FDA finds a direction to separate the class means while minimizing the total covariance of the training samples. Therefore, two quantities need to be defined, which are: 1) the between class scatter $S_b = (\vec{m}_2 - \vec{m}_1)(\vec{m}_2 - \vec{m}_1)^T$: measuring the difference between two classes; and 2) the within class scatter $S_w = \sum_{i=1}^{N}(\vec{x}_i - \vec{m})(\vec{x}_i - \vec{m})^2 = N\Sigma$: the variance of all training samples. The projection direction $\vec{w}$ maximizes

$$\left[\max_{\vec{w}}\; J_{FDA}(\vec{w}) = \frac{\vec{w}^T S_b \vec{w}}{\vec{w}^T S_w \vec{w}}\right]. \tag{4.48}$$

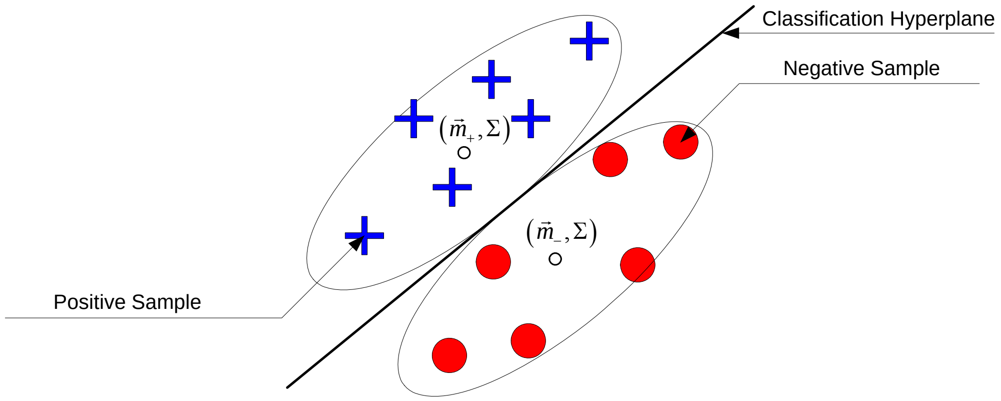


Figure 4.8. FDA separates positive samples from negative samples by maximizing the symmetric Kullback–Leibler divergence between two classes under the assumption that the two classes share the same covariance matrix.

This problem is simplified as

$$\left[\max_{\vec{w}}\; J_{FDA}(\vec{w}) = \frac{\left\|\vec{w}^T(\vec{m}_+ - \vec{m}_-)\right\|}{\sqrt{\vec{w}^T\Sigma\vec{w}}}\right]. \tag{4.49}$$

According to Chapter 2, we know this procedure is equivalent to maximizing the symmetric Kullback–Leibler divergence (KLD) between positive and negative samples with identical covariances in the projected subspace, so that positive samples are separated from negative samples. Based on the definition of FDA, we know FDA is a special case of the linear discriminant analysis (LDA).

The decision function in FDA is $y(\vec{x}) = \text{sign}\left[\vec{w}^T\vec{x}+b\right]$, where $\vec{w}$ is the eigenvector of $\Sigma^{-1}(\vec{m}_+ - \vec{m}_-)(\vec{m}_+ - \vec{m}_-)^T$ associated with the largest eigenvalue and the bias b is calculated by

$$b = \frac{N_- - N_+ - (N_+\vec{m}_+ + N_-\vec{m}_-)^T \vec{w}}{N_- + N_+}. \tag{4.50}$$

The significance [30][39] of FDA is: FDA is Bayes optimal when the two classes are Gaussian distributed with identical covariances.

When objects are represented by tensors, we need to vectorize the tensors to vectors to match the input requirments in FDA. When training samples are limited, the vectorization will be a disaster for FDA. This is because S_w and S_b are both singular. To reduce this problem, we propose the tensor extension of FDA, i.e., tensor FDA (TFDA). TFDA is a combination of FDA and STL. Moreover, TFDA is a special case of the previous proposed general tensor discriminant analysis (GTDA).

Suppose we have training samples $\mathbf{X}_i \in R^{L_1 \times L_2 \times \cdots L_M}$ ($1 \le i \le N$) and their corresponding class labels $y_i \in \{+1,-1\}$. The mean of the positive training samples is $\mathbf{M}_+ = (1/N_+)\sum_{i=1}^N\left[\text{I}(y_i=+1)\mathbf{X}_i\right]$; the mean of the negative training samples is $\mathbf{M}_- = (1/N_-)\sum_{i=1}^N\left[\text{I}(y_i=-1)\mathbf{X}_i\right]$; the mean of all training samples is $\mathbf{M} = (1/N)\sum_{i=1}^N \mathbf{X}_i$; and N_+ (N_-) is the number of positive (negative) samples.

The decision function is defined by $y(\mathbf{X}) = \text{sign}\left[\mathbf{X}\prod_{k=1}^{M} \times_k \vec{w}_k + b\right]$, where the projection vectors $\vec{w}_k \in R^{L_k}$ ($1 \le k \le M$) and the bias b in TFDA are obtained from

$$\left[\max_{\vec{w}_k|_{k=1}^M} J_{TFDA}\left(\vec{w}_k|_{k=1}^M\right) = \frac{\left\|(\mathbf{M}_+ - \mathbf{M}_-)\prod_{k=1}^{M}\times_k \vec{w}_k\right\|^2}{\sum_{i=1}^{N}\left\|(\mathbf{X}_i - \mathbf{M})\prod_{k=1}^{M}\times_k \vec{w}_k\right\|^2}\right], \tag{4.51}$$

The formula is obtained directly from (3.31) and (3.32). Similar to GTDA, there is no closed form solution for TFDA. The alternating projection is applied to obtain the solution for TFDA and we need to replace the Step 4 in Table 4.1 by the following optimization problem,

$$\left[\max_{\vec{w}_j} \quad J_{TFDA}\left(\vec{w}_j\right) = \frac{\left\| \vec{w}_j^T \left[\left(\mathbf{M}_+ - \mathbf{M}_- \right) \bar{\times}_j \vec{w}_j \right] \right\|^2}{\sum_{i=1}^{N} \left\| \vec{w}_j^T \left[\left(\mathbf{X}_i - \mathbf{M} \right) \bar{\times}_j \vec{w}_j \right] \right\|^2} \right]. \tag{4.52}$$

This problem is the standard FDA. When we have the projection vectors $\vec{w}_k |_{k=1}^{M}$, we can obtain the bias b from

$$b = \frac{N_- - N_+ - \left(N_+ \mathbf{M}_+ + N_- \mathbf{M}_- \right) \prod_{k=1}^{M} \times_k \vec{w}_k}{N_- + N_+}. \tag{4.53}$$

Distance Metric Learning vs. Multiple Distance Metrics Learning

Weinberger et al. [169] proposed the distance metric learning (DML) to learn a metric for *k*–nearest–neighbor (*k*NN) classification. The motivation of DML is simple because the performance of *k*NN is only related to the metric used for dissimilarity measure. In traditional *k*NN, the Euclidean metric fails to capture the statistical charateristics of training samples. In DML, the metric is obtained such that "*k–nearest neighbors always belong to the same class while examples from different classes are separated by a large margin*". DML is defined by

$$\left[\begin{array}{ll} \min\limits_{\substack{\Sigma, \xi_{ijl} \\ 1 \le i,j,l \le N}} & J_{DML}\left(\Sigma, \xi_{ijl}\right) = \sum_{i=1}^{N} \sum_{j=1}^{N} \eta_{ij} \left(\vec{x}_i - \vec{x}_j \right)^T \Sigma \left(\vec{x}_i - \vec{x}_j \right) + c \sum_{i=1}^{N} \sum_{j=1}^{N} \eta_{ij} \left(1 - y_{il} \right) \xi_{ijl} \\ & \left(\vec{x}_i - \vec{x}_l \right)^T \Sigma \left(\vec{x}_i - \vec{x}_l \right) - \left(\vec{x}_i - \vec{x}_j \right)^T \Sigma \left(\vec{x}_i - \vec{x}_j \right) \ge 1 - \xi_{ijl}, \quad 1 \le i,j,l \le N \\ \text{s.t.} & \xi_{ijl} \ge 0, \qquad\qquad 1 \le i,j,l \le N \\ & \Sigma \ge 0 \end{array} \right] \tag{4.54}$$

where $\eta_{ij} = 1$ and $y_{ij} = 1$ mean that $\vec{x}_i$ and $\vec{x}_j$ have the same class label, otherwise 0. The constraint function $\Sigma \ge 0$ indicates that the maxtrix Σ is required to be positive semidefinite, so the problem is an SDP. From the learnt distance metric Σ, it is direct to have the linear transformation matrix by decomposing $\Sigma = W^T W$.

The optimization problem defined in (4.54) is equivalent to

$$\left[\begin{array}{lll} \min\limits_{\substack{\Sigma,\xi_{ijl}\\ 1\le i,j,l\le N}} & J_{DML}\left(\Sigma,\xi_{ijl}\right)=\operatorname{tr}\left(A^T\Sigma A\right)+c\sum_{i=1}^{N}\sum_{j=1}^{N}\eta_{ij}\left(1-y_{il}\right)\xi_{ijl} & \\ & B_{ijl}^T\Sigma B_{ijl}\ge 1-\xi_{ijl}, & 1\le i,j,l\le N \\ \text{s.t.} & \xi_{ijl}\ge 0, & 1\le i,j,l\le N \\ & \Sigma\ge 0 & \end{array}\right] \tag{4.55}$$

where $A=\left[\sqrt{\eta_{ij}}\left(\vec{x}_i-\vec{x}_j\right)\right]_{L\times N^2}$ ($1\le i,j\le N$) and $B_{ijl}=\left[\vec{x}_i-\vec{x}_l,\vec{x}_j-\vec{x}_i\right]_{L\times 2}$.

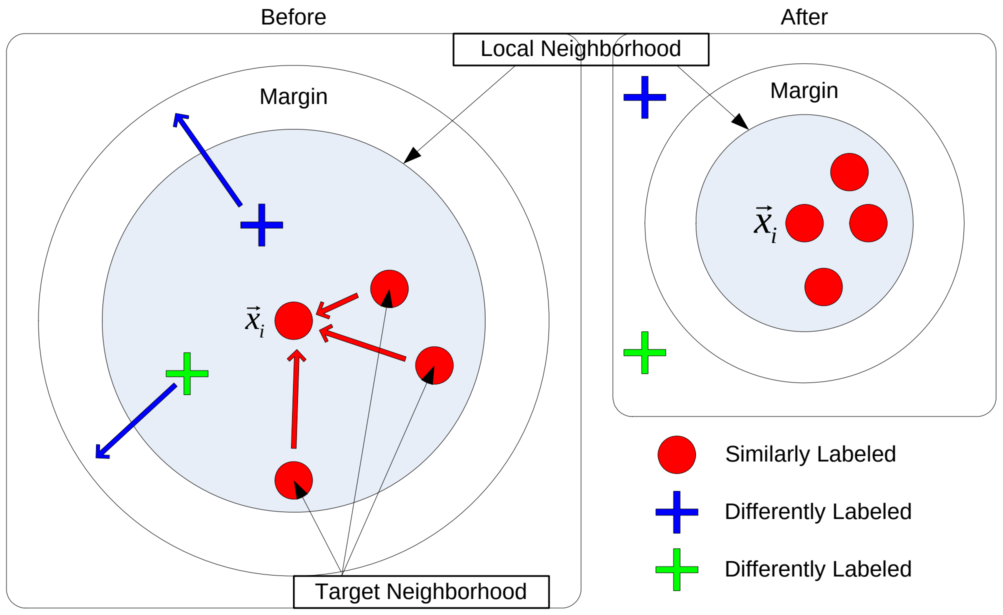


Figure 4.9. DML obtains a metric, such that "*k–nearest neighbors always belong to the same class while examples from different classes are separated by a large margin*".

Suppose we have training samples $\mathbf{X}_i\in R^{L_1\times L_2\times\cdots L_M}$ ($1\le i\le N$) and their corresponding class labels $y_i\in\{1,2,\cdots,n\}$. The multiple distance metric learning (MDML) learns M metrics $\Sigma_k=W_k^TW_k$ ($1\le k\le M$) for M–th order tensors $\mathbf{X}_i|_{i=1}^{N}$ to make the samples, which have the same (different) labels, be as close (far) as possible. The MDML is defined as

$$\left[\begin{array}{l} \min\limits_{\substack{W_k|_{k=1}^M,\xi_{ijl}\\ 1\le i,j,l\le N}} J_{MDML}\left(W_k|_{k=1}^M,\xi_{ijl}|_{1\le i,j,l\le N}\right)=\left[\begin{array}{l}\sum\limits_{i=1}^N\sum\limits_{j=1}^N\eta_{ij}\left\|\left(\mathbf{X}_i-\mathbf{X}_j\right)\prod\limits_{k=1}^M\times_k W_k\right\|_{Fro}^2\\ +c\sum\limits_{i=1}^N\sum\limits_{j=1}^N\eta_{ij}\left(1-y_{il}\right)\xi_{ijl}\end{array}\right]\\ \qquad\left\|\left(\mathbf{X}_i-\mathbf{X}_l\right)\prod\limits_{k=1}^M\times_k W_k\right\|_{Fro}^2-\left\|\left(\mathbf{X}_i-\mathbf{X}_j\right)\prod\limits_{k=1}^M\times_k W_k\right\|_{Fro}^2\ge 1-\xi_{ijl}\\ \text{s.t.}\quad \xi_{ijl}\ge 0,\qquad 1\le i,j,l\le N\\ \qquad\; W_k^TW_k\ge 0,\quad 1\le k\le M\end{array}\right]. \tag{4.56}$$

As described in the STL framework, there is also no closed form solution for MDML. The alternating projection method is applied to obtain the solution for MDML and we need to replace the Step 4 in Table 4.1 by the following optimization problem,

$$\left[\begin{array}{l} \min\limits_{\substack{W_p,\xi_{ijl}\\ 1\le i,j,l\le N}} J_{MDML}\left(W_p,\xi_{ijl}|_{1\le i,j,l\le N}\right)=\operatorname{tr}\left(A_p^T\Sigma_pA_p\right)+c\sum\limits_{i=1}^N\sum\limits_{j=1}^N\eta_{ij}\left(1-y_{il}\right)\xi_{ijl}\\ \qquad\left\|\left(\mathbf{X}_i-\mathbf{X}_l\right)\prod\limits_{k=1}^M\times_k W_k\right\|_{Fro}^2-\left\|\left(\mathbf{X}_i-\mathbf{X}_j\right)\prod\limits_{k=1}^M\times_k W_k\right\|_{Fro}^2\ge 1-\xi_{ijl}\\ \text{s.t.}\quad \xi_{ijl}\ge 0,\qquad 1\le i,j,l\le N\\ \qquad\; W_k^TW_k\ge 0,\quad 1\le k\le M\end{array}\right]. \tag{4.57}$$

Here,

$$A_p=\sum_{i=1}^N\sum_{j=1}^N\sqrt{\eta_{ij}}\operatorname{mat}_p\left(\left(\mathbf{X}_i-\mathbf{X}_j\right)\bar{\times}_pW_p\right), \tag{4.58}$$

and

$$B_{ijl;p}=\operatorname{mat}_p\left(\left(\mathbf{X}_j-\mathbf{X}_l\right)\bar{\times}_pW_p\right). \tag{4.59}$$

This is because

$$\left\|\left(\mathbf{X}_i-\mathbf{X}_l\right)\prod_{k=1}^M\times_k W_k\right\|_{Fro}^2=\operatorname{tr}\left(\operatorname{mat}_j^T\left(\left(\mathbf{X}_i-\mathbf{X}_l\right)\bar{\times}_jW_j\right)\Sigma_j\operatorname{mat}_j\left(\left(\mathbf{X}_i-\mathbf{X}_l\right)\bar{\times}_jW_j\right)\right).$$

Deduction:

$$\left\|\left(\mathbf{X}_i-\mathbf{X}_l\right)\prod_{k=1}^M\times_k W_k\right\|_{Fro}^2$$

$$=\left\llbracket\left(\left(\mathbf{X}_i-\mathbf{X}_l\right)\prod_{k=1}^M\times_k W_k\right)\otimes\left(\left(\mathbf{X}_i-\mathbf{X}_l\right)\prod_{k=1}^M\times_k W_k\right);(1:M)(1:M)\right\rrbracket$$

$$=\operatorname{tr}\left\llbracket\left(\left[\left(\mathbf{X}_i-\mathbf{X}_l\right)\bar{\times}_jW_j\right]\times_jW_j\right)\otimes\left(\left[\left(\mathbf{X}_i-\mathbf{X}_l\right)\bar{\times}_jW_j\right]\times_jW_j\right);(1:M)(1:M)\right\rrbracket$$

$$=\operatorname{tr}\left(W_j\left\llbracket\left[\left(\mathbf{X}_i-\mathbf{X}_l\right)\bar{\times}_jW_j\right]\otimes\left[\left(\mathbf{X}_i-\mathbf{X}_l\right)\bar{\times}_jW_j\right];(\bar{j})(\bar{j})\right\rrbracket W_j^T\right)$$

$$= \operatorname{tr}\left(W_j \operatorname{mat}_j\left(\left(\mathbf{X}_i - \mathbf{X}_l\right) \overline{\times}_j W_j\right) \operatorname{mat}_j^T\left(\left(\mathbf{X}_i - \mathbf{X}_l\right) \overline{\times}_j W_j\right) W_j^T\right)$$

$$= \operatorname{tr}\left(\operatorname{mat}_j^T\left(\left(\mathbf{X}_i - \mathbf{X}_l\right) \overline{\times}_j W_j\right) W_j^T W_j \operatorname{mat}_j\left(\left(\mathbf{X}_i - \mathbf{X}_l\right) \overline{\times}_j W_j\right)\right)$$

$$= \operatorname{tr}\left(\operatorname{mat}_j^T\left(\left(\mathbf{X}_i - \mathbf{X}_l\right) \overline{\times}_j W_j\right) \Sigma_j \operatorname{mat}_j\left(\left(\mathbf{X}_i - \mathbf{X}_l\right) \overline{\times}_j W_j\right)\right).$$

This problem defined in (4.57) is the standard DML.

Iterative Feature Extraction Model based on Supervised Tensor Learning

The iterative feature extraction model (IFEM) based on STL is an extension of the STL framework for feature extraction and its procedure is similar to the recursive rank one tensor approximation developed by Shashua and Levin in [132].

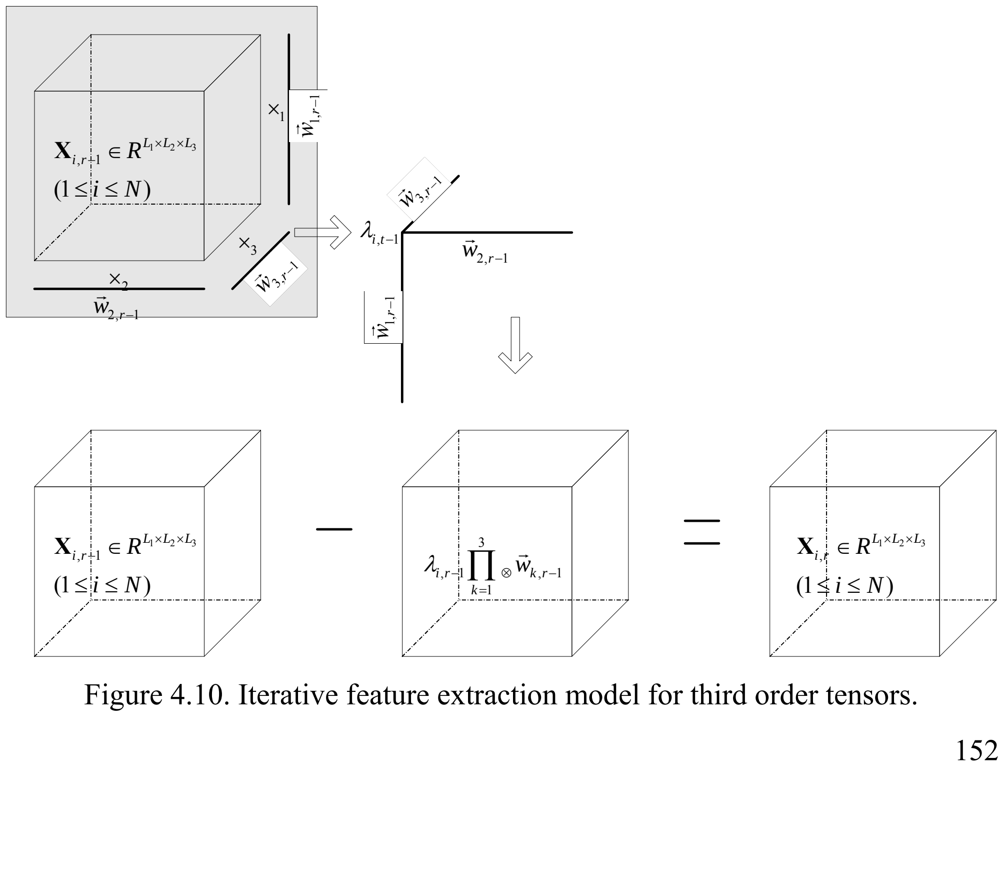
Suppose we have training samples $\mathbf{X}_i \in R^{L_1 \times L_2 \times \cdots L_M}$ ($1 \le i \le N$) and their corresponding class labels $y_i \in \{+1, -1\}$. IFEM is defined by

$$\mathbf{X}_{i,r} = \mathbf{X}_{i,r-1} - \lambda_{i,r-1} \prod_{k=1}^{M} \times_k \vec{w}_{k,r-1} \tag{4.60}$$

$$\lambda_{i,r-1} = \mathbf{X}_{i,r-1} \prod_{k=1}^{M} \times_k \left(\vec{w}_{k,r-1}\right)^T \tag{4.61}$$

$$\begin{bmatrix} \min\limits_{\vec{w}_{k,r}|_{k=1}^{M}, b, \vec{\xi}} & f\left(\vec{w}_{k,r}|_{k=1}^{M}, b, \vec{\xi}\right) \\ \text{s.t.} & y_i c_i \left(\mathbf{X}_{i,r} \prod_{k=1}^{M} \times_k \vec{w}_{k,r} + b \right) \ge \xi_i, \quad 1 \le i \le N \end{bmatrix} \tag{4.62}$$

where $\mathbf{X}_{i,1} = \mathbf{X}_i$ and $\lambda_{i,0} = 0$. The $\lambda_{i,r}|_{r=1}^{R}$ (R is the number of extracted features in IFEM) is used to represent the original tensor $\mathbf{X}_i$.



$\mathbf{X}_{i,r-1} \in R^{L_1 \times L_2 \times L_3}$
$(1 \le i \le N)$
$\times_1$
$\vec{w}_{1,r-1}$
$\times_2$
$\vec{w}_{2,r-1}$
$\times_3$
$\vec{w}_{3,r-1}$
$\lambda_{i,t-1}$
$\vec{w}_{3,r-1}$
$\vec{w}_{2,r-1}$
$\vec{w}_{1,r-1}$
$\mathbf{X}_{i,r-1} \in R^{L_1 \times L_2 \times L_3}$
$(1 \le i \le N)$
$-$
$\lambda_{i,r-1} \prod_{k=1}^{3} \otimes \vec{w}_{k,r-1}$
$=$
$\mathbf{X}_{i,r} \in R^{L_1 \times L_2 \times L_3}$
$(1 \le i \le N)$

Figure 4.10. Iterative feature extraction model for third order tensors.

From the definition of IFEM, which is defined by Eqs. (4.60), (4.61), and (4.62), we know that IFEM can be calculated by a greedy approach. The calculation of $\mathbf{X}_{i,r}\,|_{i=1}^{N}$ is based on the given $\mathbf{X}_{i,r-1}\,|_{i=1}^{N}$ and $\vec{w}_{k,r-1}\,|_{k=1}^{M}$. With the given $\mathbf{X}_{i,r-1}\,|_{i=1}^{N}$ and $\vec{w}_{k,r-1}\,|_{k=1}^{M}$, we can calculate $\lambda_{i,r-1}$ via (4.61). The projection vectors $\vec{w}_{k,r}\,|_{k=1}^{M}$ can be obtained by optimizing (4.62) through the alternating projection method in Table 4.1. The flowchart of the algorithm for feature extraction for third order tensors is illustrated in Figure 4.10.

With IFEM, we can obtain $\vec{w}_{k,r}\,|_{1\le k\le M}^{1\le r\le R}$ iteratively. The coordinate values $\lambda_{i,r}\,|_{r=1}^{R}$ can represent the original tensor $\mathbf{X}_i$. For example, in nearest neighbor based recognition, the prototype tensor $\mathbf{X}_p$ for each individual class in the database and the testing tensor $\mathbf{X}_t$ to be classified are projected onto the bases to get the prototype vector $\lambda_{p,r}\,|_{r=1}^{R}$ and the testing vector $\lambda_{t,r}\,|_{r=1}^{R}$. The testing tensor class is found by minimizing the Euclidean distance $\varepsilon=\sqrt{\sum_{r=1}^{R}\left(\lambda_{t,r}-\lambda_{p,r}\right)^2}$ over p.

As an example, we develop the tensor rank one discriminant analysis (TR1DA) by combining IFEM with differential scatter discriminant criterion (DSDC) described in Chapter 2.

TR1DA deals with the multiple classes classification problem. Suppose: there are *N* training samples $\mathbf{X}_{i;j}\in R^{L_1\times L_2\times\cdots L_M}$. The $\mathbf{X}_{i;j,r}$ is the *j*th ($1\le j\le N_i$) training sample in the *i*th ($1\le i\le C$) class for the *r*th iteration for feature extraction. If *r* equals to 1, we have $\mathbf{X}_{i;j,1}=\mathbf{X}_{i;j}$. The *i*th class mean tensor in the *r*th iteration is $\mathbf{M}_{i,r}=\frac{1}{N_i}\sum_{j=1}^{N_i}\mathbf{X}_{i;j,r}$ and the total mean tensor in the *r*th iteration is $\mathbf{M}_r=\frac{1}{C}\sum_{i=1}^{C}\frac{1}{N_i}\sum_{i=1}^{N_i}\mathbf{X}_{i;j,r}=\sum_{i=1}^{C}\frac{N_i}{N}\mathbf{M}_{i,r}$. The *k*th projection vector in the *r*th iteration is defined by $\vec{w}_{k,r}$. With the given $\mathbf{X}_{i;j,r}$ and $\vec{w}_{k,r}$, the $(r+1)$th iteration for feature extraction in TR1DA is defined by

$$\mathbf{X}_{i;j,r+1}=\mathbf{X}_{i;j,r}-\lambda_{i;j,r}\prod_{k=1}^{M}{}_{\otimes}\vec{w}_{k,r} \tag{4.63}$$

$$\lambda_{i;j,r}=\mathbf{X}_{i;j,r}\prod_{k=1}^{M}{}_{\times_k}\vec{w}_{k,r}^{T} \tag{4.64}$$

$$\vec{w}_{k,r}^{*}|_{k=1}^{M} = \arg\max_{\vec{w}_{k,r}|_{k=1}^{M}} \begin{pmatrix} \sum_{i=1}^{C} N_i \left| \left(\mathbf{M}_{i,r} - \mathbf{M}_r\right) \prod_{k=1}^{M} \times_k \vec{w}_{k,r} \right|^2 \\ -\zeta_r \sum_{i=1}^{C} \sum_{j=1}^{N_i} \left| \left(\mathbf{X}_{i;j,r} - \mathbf{M}_{i,r}\right) \prod_{k=1}^{M} \times_k \vec{w}_{k,r} \right|^2 \end{pmatrix}. \quad (4.65)$$

Table 4.2. Alternating Projection for the Tensor Rank One Discriminant Analysis

Input: Training samples $\mathbf{X}_{i;j} \in R^{L_1 \times L_2 \times \ldots \times L_M}$ $(1 \le i \le c,\ 1 \le j \le n_i)$, the number R of rank one tensors allowed in TR1DA, and tuning parameters ζ_r, $1 \le r \le R$ in TR1DA.

Output: Projection vectors $\left\{ \vec{w}_{k,r} |_{k=1}^{M}, \vec{w}_{k,r} \in R^{L_k} \right\}$ and scalars $\lambda_{i;j,r}$, $1 \le r \le R$.

```
1.   For $r = 1$ to $R$
2.       Set $\vec{w}_{k,r} |_{k=1}^{M}$ be equal to random unit vectors.
3.       Calculate the class mean tensor $\mathbf{M}_i = \left(1/N_i\right) \sum_{j=1}^{N_i} \mathbf{X}_{i;j}$;
         Calculate the total mean tensor $\mathbf{M} = \sum_{i=1}^{C} \left(N_i / N\right) \mathbf{M}_i$;
4.       Carry out steps 5–10 iteratively until convergence.
5.       For $k = 1$ to $M$
6.           Calculate $B = \sum_{i=1}^{C} \left[ N_i \left( \left(\mathbf{M}_i - \mathbf{M}\right) \bar{\times}_k \vec{w}_{k,r}^T \right) \otimes \left( \left(\mathbf{M}_i - \mathbf{M}\right) \bar{\times}_k \vec{w}_{k,r}^T \right) \right]$;
             Calculate $W = \sum_{i=1}^{C} \sum_{j=1}^{N_i} \left[ \left( \left(\mathbf{X}_{i;j} - \mathbf{M}_i\right) \bar{\times}_k \vec{w}_{k,r}^T \right) \otimes \left( \left(\mathbf{X}_{i;j} - \mathbf{M}_i\right) \bar{\times}_k \vec{w}_{k,r}^T \right) \right]$;
7.           Calculate the eigenvector $\vec{h}$ of $B - \zeta^l W$ associated with the largest eigenvalue;
8.           Assignment: $\vec{w}_{k,r} \leftarrow \vec{h}$;
9.       End
10.      Convergence checking: if $\left\| \vec{w}_{k,r,t}^T \vec{w}_{k,r,t-1} \right| - 1 \Big/ \left| \vec{w}_{k,r,t-1} \right|_0 \le \varepsilon$ $(\varepsilon = 10^{-6})$ for all modes, the calculated $\vec{w}_{k,r}$ has converged. Here $\vec{w}_{k,r,t}$ is the current projection vector and $\vec{w}_{k,r,t-1}$ is the previous projection vector.
11.      Assignment: $\lambda_{i;j,r} \leftarrow \mathbf{X}_{i;j} \prod_{k=1}^{M} \times_k \vec{w}_{k,r}^T$;
12.      Assignment: $\mathbf{X}_{i;j} \leftarrow \mathbf{X}_{i;j} - \lambda_{i;j,r} \prod_{k=1}^{M} \otimes \vec{w}_{k,r}$;
13.  End
```

The algorithm is given in Table 4.2.

The time complexity of PCA (LDA) is $O\left(\left(\prod_{k=1}^{M} L_k\right)^3\right)$ when training samples $\mathbf{X}$ belong to $R^{L_1 \times L_2 \times \cdots \times L_M}$. The time complexity of TR1DA is $O\left(\sum_{r=1}^{R} T_r \sum_{k=1}^{M} L_k\right)$, where T_r is the number of iterations to make TR1DA converge for the rth feature extraction procedure (in our experiments for human gait recognition [143][145], T_r is about 20). The space complexity of PCA (LDA) is $O\left(\left(\prod_{k=1}^{M} L_k\right)^2\right)$. The space complexity of TR1DA is $O\left(\prod_{k=1}^{M} L_k\right)$. This indicates that the time complexity and the space complexity of feature extraction are reduced by working directly with tensor data rather than vectorizing the data and applying PCA (LDA).

Experiments

In this Section, we provide two experiments to demonstrate that STL and IFEM are powerful tools for classification and feature extraction. For STL, we implement TMPM for image classification. For IFEM, we implement TR1DA for the elapsed time problem in human gait recognition.

TMPM for Image Classification

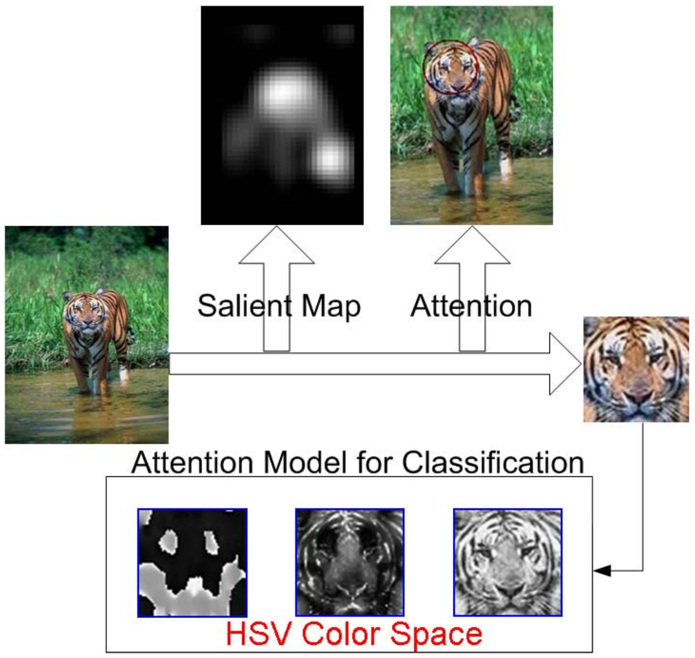


Figure 4.11. Attention model for image representation.

To classify images into groups based on their semantic contents is very important and challenging. The simplest classification is binary and a hierarchical structure can be built from a series of binary classifiers. If we have semantic classification then image databases would be easier to manage [121][125]. The image semantic classification is also of great help for many applications.

In this STL based classification experiment, two groups of images are separated from each other by a trained TMPM, which is a generalized learning machine with the STL framework. The input (representing features) of TMPM is the region of interest (ROI) within an image, which are extracted by the attention model in [56][57][140] and represented as a third order tensor.

The attention model [56][57] is capable of reproducing human performances for a number of pop–out tasks [157]. In other words, when a target is different from its surroundings by its unique orientation, color, intensity, or size, it is always the first attentive location and easy to be noticed by an observer. Therefore, it is reasonable to utilize the attention model based ROI to describe the semantic information of an image.

Figure 4.12. Example images from the tiger category.

As shown in Figure 4.11, representing an attention region from an image consists of several steps: 1) extracting the salient map as introduced by Itti *et al.* [56][57];

2) finding the most attentive region, whose center has the largest value in the salient map; 3) extracting the attention region by a square, i.e., ROI, in size of 64×64; and 4) finally, representing this ROI in the hue, saturation, and value (HSV) perceptual color space. We have a third order tensor for the image representation.

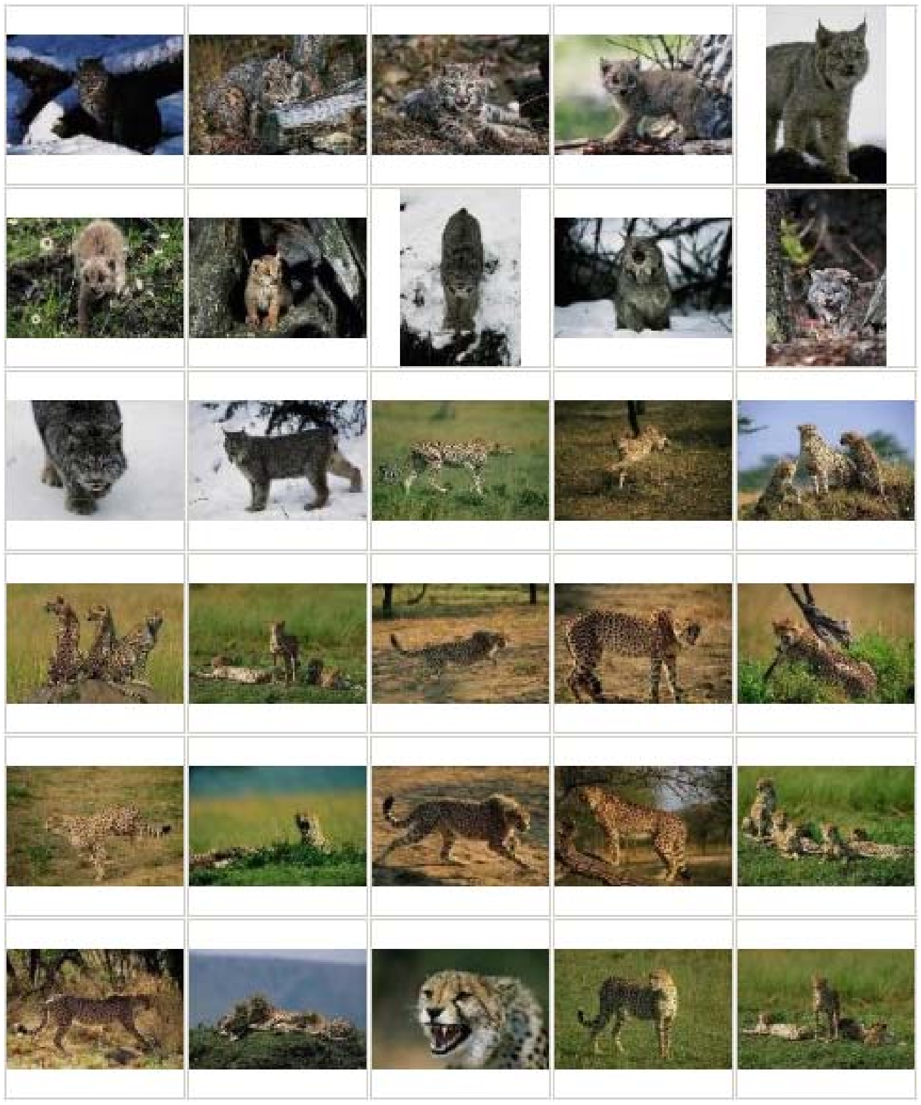


Figure 4.13. Example images from the leopard category.

Note that although we only select a small region from an image, the size of the extracted third order tensor is already as large as $64 \times 64 \times 3$. If we vectorize this tensor, the dimension of the vector will be 12288. The sizes of training samples are only of hundreds, which is much smaller than 12288. Therefore, the small sample size (SSS) problem always arises. On the contrary, our proposed tensor oriented supervised learning scheme can avoid this problem directly and at the same time represent the ROIs much more naturally.

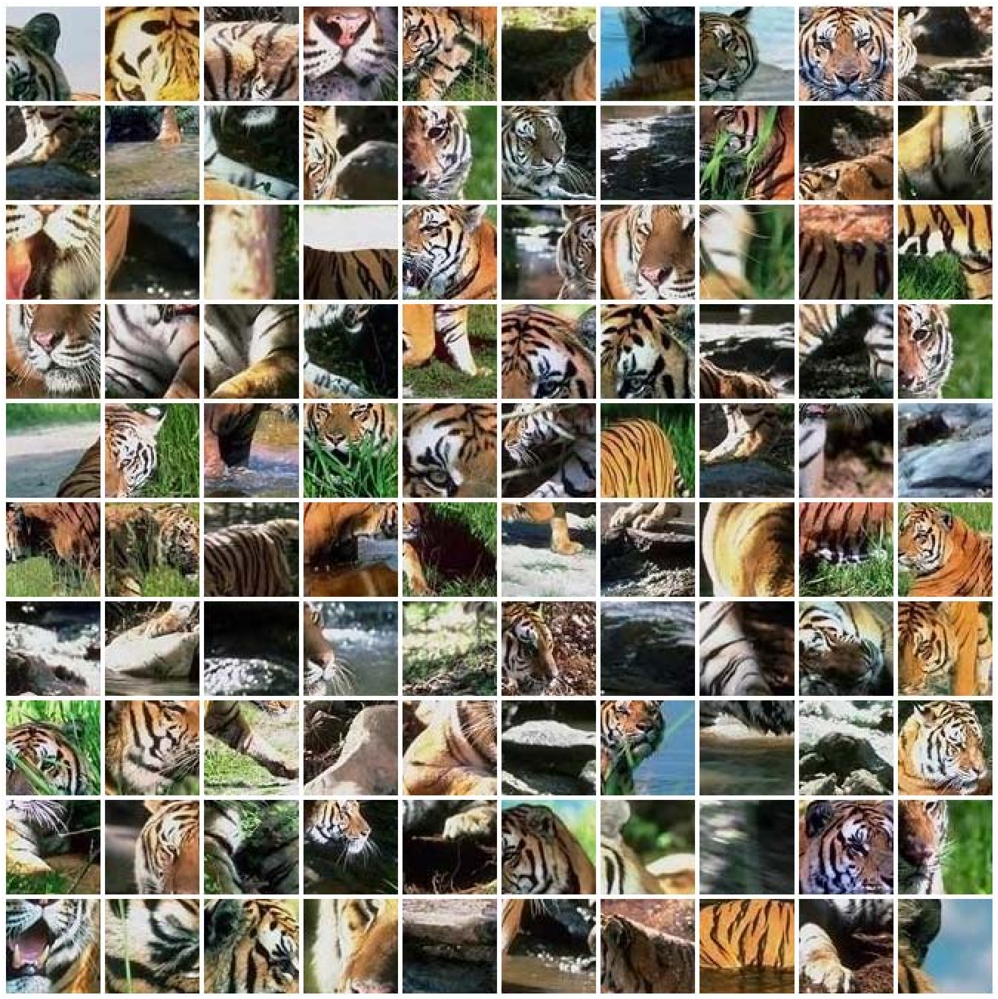


Figure 4.14. One hundred ROIs in the tiger category.

The training set and the testing set for the following experiments are built upon the Corel photo gallery [164], from which 100 images are selected for each of the two categories. Examples are shown in Figure 4.12 and Figure 4.13. These 200 images are then processed to extract the third tensor attention features for TMPM as shown in Figure 4.14 and Figure 4.15.

We choose the “Tiger” category shown in Figure 4.12 and the “Leopard” category shown in Figure 4.13 for this binary classification experiment since it is a very difficult task for a machine to distinguish them, although a human being can easily differentiate between a tiger and a leopard. Basically, the characteristics of a classifier cannot be examined in detail if the classification problem is very straightforward, for example, classifying grassland pictures from blood pictures. The “Tiger” and “Leopard” classification is carried out in this Section. We choose the top *N* images as training sets according to the image IDs, while all remaining images are used to form the corresponding testing set.

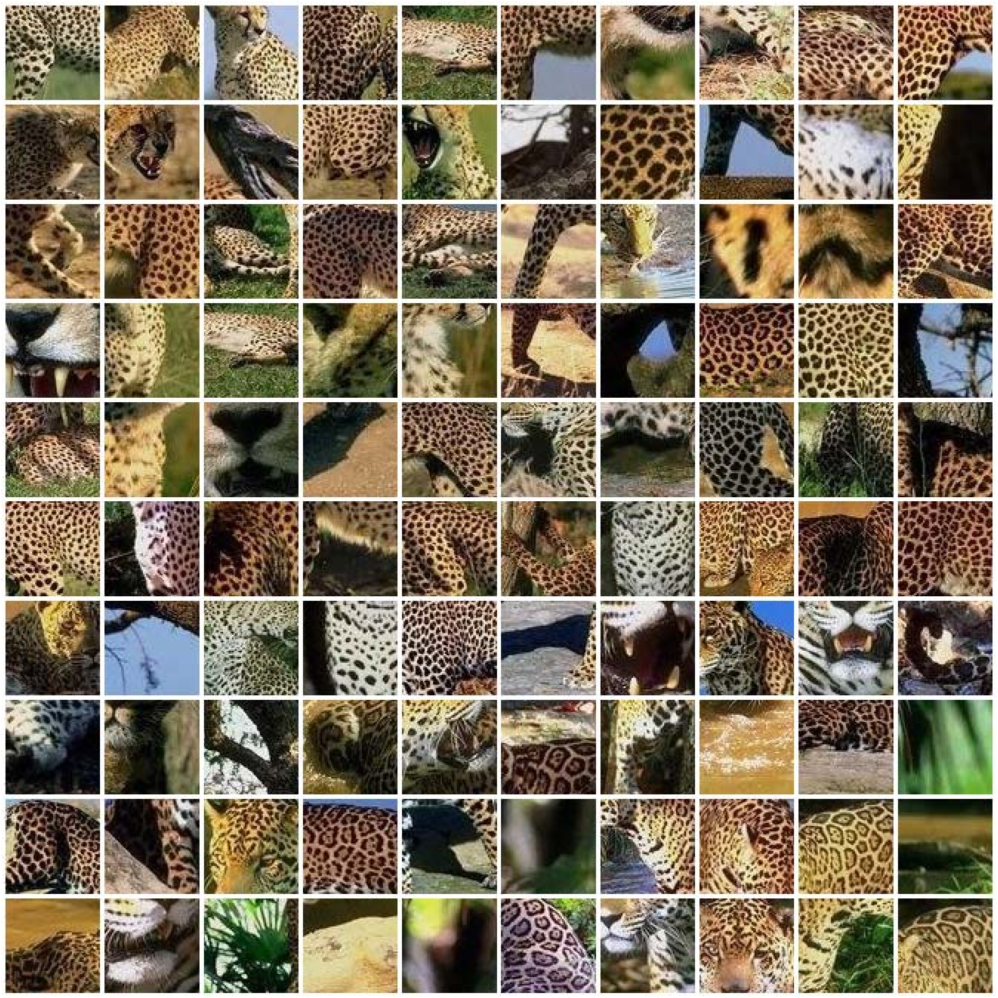


Figure 4.15. One hundred ROIs in the leopard category.

In our experiments, the third order tensor attention ROIs can mostly be found correctly from images. Some successful results, respectively extracted automatically from the “Tiger” category and the “Leopard” category, are shown in Figure 4.14 and Figure 4.15. By this we mean that, the underlying data structures are kept well for the next classification step. However, we should note that the attention model sometimes cannot depict the semantic information of an image. This is mainly because the attention model always locates a region that is different from its surroundings and thus might be “cheated” when some complex or bright background exists. Some unsuccessful ROIs can also be found from Figure 4.14 and Figure 4.15. It should be emphasized that to keep the following comparative experiments fair and automatic, these wrongly extracted ROIs are included in training samples.

We carried out the binary classification ("Tiger" and "Leopard") experiments upon the above training and testing sets. The proposed TMPM is compared with MPM. The experimental results are shown in Table 4.3. Error rates for both training and testing are reported according to the increasing size of the training set (STS) from 5 to 30.

Table 4.3. TMPM vs. MPM

	Training Error Rate		Testing Error Rate	
STS	TMPM	MPM	TMPM	MPM
5	0.0000	0.4000	0.4600	0.5050
10	0.0000	0.5000	0.4250	0.4900
15	0.0667	0.4667	0.3250	0.4150
20	0.0500	0.5000	0.2350	0.4800
25	0.0600	0.4800	0.2400	0.4650
30	0.1167	0.5000	0.2550	0.4600

From the training error rates in Table 4.3, it can be seen that the traditional method (MPM) cannot learn a suitable model for classification when the size of the training set is much smaller than the dimension of the feature space. It shows that the proposed TMPM algorithm is more effective than MPM at representing the intrinsic discriminative information (in the form of the third order ROIs). TMPM learns a better classification model for future data classification than MPM and thus has a satisfactory performance on the testing set. It is also observed that the TMPM error rate is a decreasing function of the size of the training set. This is consistent with statistical learning theory.

We also evaluate TMPM as a sample algorithm of the proposed STL framework. Two important issues in machine learning are studied, namely, the training stage convergence property and the insensitiveness to the initial values.

Figure 4.16 shows that as the training stage of TMPM converges efficiently by the alternating projection method. Usually, twenty iterations are enough to achieve convergence.

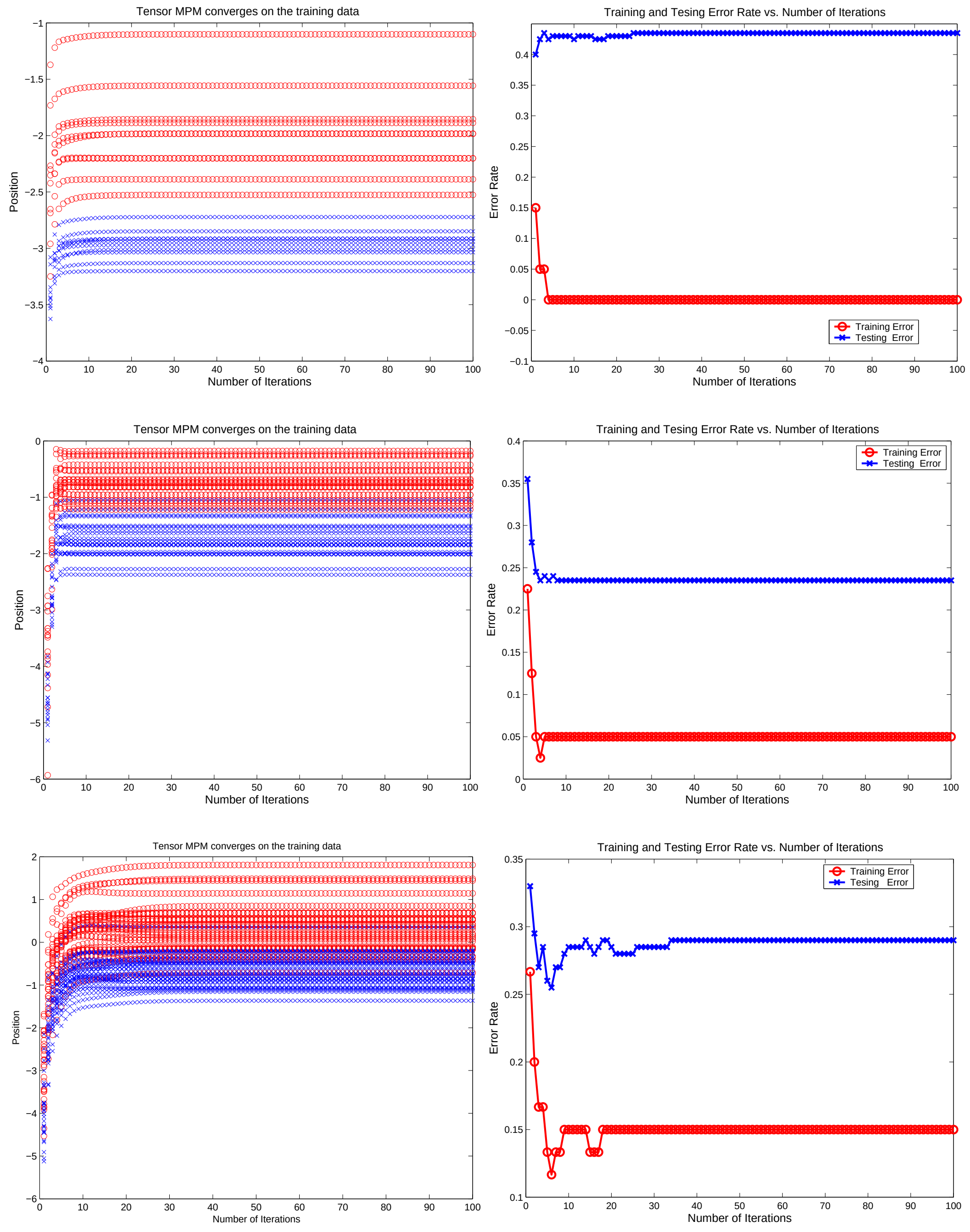


Figure 4.16. TMPM converges effectively.

Three sub–figures in the left column of Figure 4.16 show tensor projected values of the original general tensors with an increasing number of learning iterations using 10, 20, and 30 training samples for each class, respectively, from top to bottom. We find that the projected values converge to stable values. Three sub–figures in the right column of Figure 4.16 show the training error rates and the testing error rates according to the increasing number of learning iterations by 10, 20, and 30 training samples for each class, respectively, from top to bottom.

Based on all sub–figures in Figure 4.16, it can be found that the training error and the testing error converge to stable values, which empirically justify the convergence of the alternating projection method for TMPM. The theoretical proof is given in the Theorem 4.1.

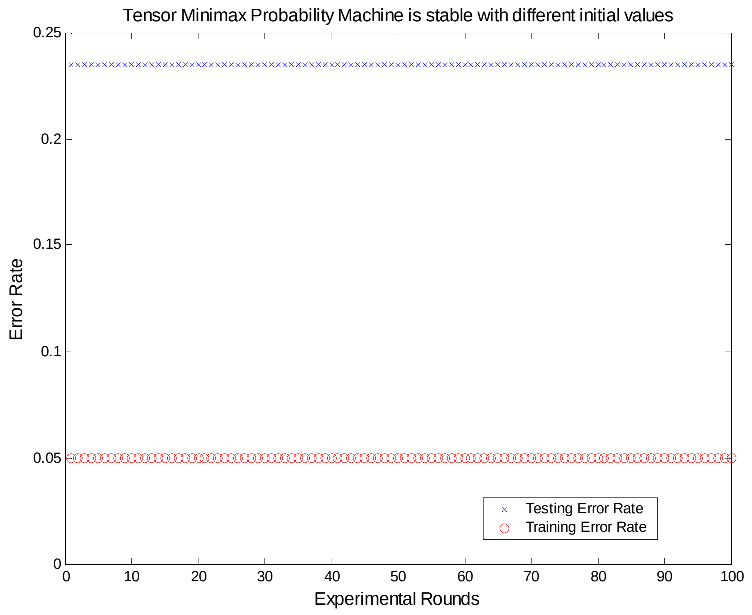


Figure 4.17. TMPM is stable with different initial values.

Many learning algorithms converge to different values when initial values are varied. This is the so–called *local minimum* problem. However, the developed TMPM does not have this *local minimum* problem, which is demonstrated by a set of experiments (with 100 different initial parameters, 10 learning iterations, and 20 training samples), as shown in Figure 4.17. The training error rates and the testing error rates are always 0.05 and 0.235, respectively. Moreover, because TMPM is a convex optimization problem, theoretically TMPM has a unique solution.

TR1DA for the Elapsed Time Problem in Gait Recognition

To study the characteristics of the proposed TR1DA, we utilize it on the elapsed time problem in human gait recognition. In this Section we first briefly introduce our experimental data (gallery and probe) sets; and then report the performance of the TR1DA algorithm and compare its performance with principal component analysis (PCA), linear discriminant analysis (LDA) and tensor rank one analysis (TR1A). The experiments provide numerical evidence for the convergence of TR1DA during the learning stage.

Research efforts in biometrics are mainly motivated by the increasing requirements of machine based automatic human authentication/authorization [71]. As a promising biometric source, human gait describes the manner of a person's walking and can be acquired at a distance. It was first analyzed in medical surgery [111][112], and then in psychology [23]. In computer vision, human motion has been studied for years [1]. An early attempt to recognize a person by the gait was probably made by Little and Boyd [86]. Since then, many efforts have been made for gait recognition [9][22][63][163]. In this Section, we focus on an appearance based model, which could be a preprocessing step for statistical models, such as Hidden Markov Model [65][66].

Our experiments are all carried out on the USF HumanID outdoor gait database [126], which consists of the following covariates: change in viewpoints (Left/L or Right/R), change in shoe types (A or B), change in walking surface (Grass/G or Concrete/C), change in carrying conditions (Briefcase/B or No Briefcase/NB), and elapsed time (May or November) between sequences being compared. The detailed description about the database is given in §3.1.

Among all five covariates, elapsed time is our main concern in this Section because human identification systems are normally required to work over long periods of time. In this thesis, the remaining three conditions are examined thoroughly, namely change in shoe types (A or B), change in walking surface (Grass or Concrete), and change in carrying conditions (carrying a Briefcase or No Briefcase). Consequently, we choose our gallery set (May, C, A, L, NB) from the May data and consider eight pattern classification problems with the test sets of CAL (Nov., C, A, L, NB), CBL (Nov., C, B, L, NB) , GAL (Nov., G, A, L, NB), GBL (Nov., G, B, L, NB), CBAL (Nov., C, A, L, BF), CBBL (Nov., C, B, L, BF), GBAL (Nov., G, A, L, BF), and GBBL (Nov., G, B, L, BF).

In this Section we empirically study the TR1DA approach in terms of accuracy, convergence during the learning phase, and impact of parameter values on accuracy. TR1DA is compared with existing methods, including PCA, LDA, and TR1A. Experiments are carried out to recognize a gait with given examples of gaits collected in the past (the elapsed time problem), which is still one of the most difficult problems in human gait recognition.

In these experiments, we use EigenGait in a PCA based method for gait recognition, FisherGait in an LDA based method [45], TR1AGait in a TR1A for gait recognition, and TR1DAGait in a TR1DA for gait recognition. Figure 4.18 shows the first ten EigenGaits, FisherGaits, TR1AGaits, and TR1DAGaits, respectively. We believe that TR1DA is a good method for classifying gaits because our experiments support this belief. First, TR1DAGait usually performs better than existing methods (EigenGait, FisherGait, and TR1AGait) in recognition tasks. Second, the training procedure for TR1DA converges within about 20 iterations. Third, it is not difficult to obtain a reasonable good performance of TR1DA by adjusting the tuning parameters ς_r ($1 \le r \le R$). This is because there is a wide range of values for ς_r, over which TR1DA achieves a good performance. In our experiments, although we extract R features, given R independent tuning parameters, we set all tuning parameters be equal to each other, in order to reduce the number of parameters. It is possible to achieve better performance if the values of the tuning parameters are allowed to be different.

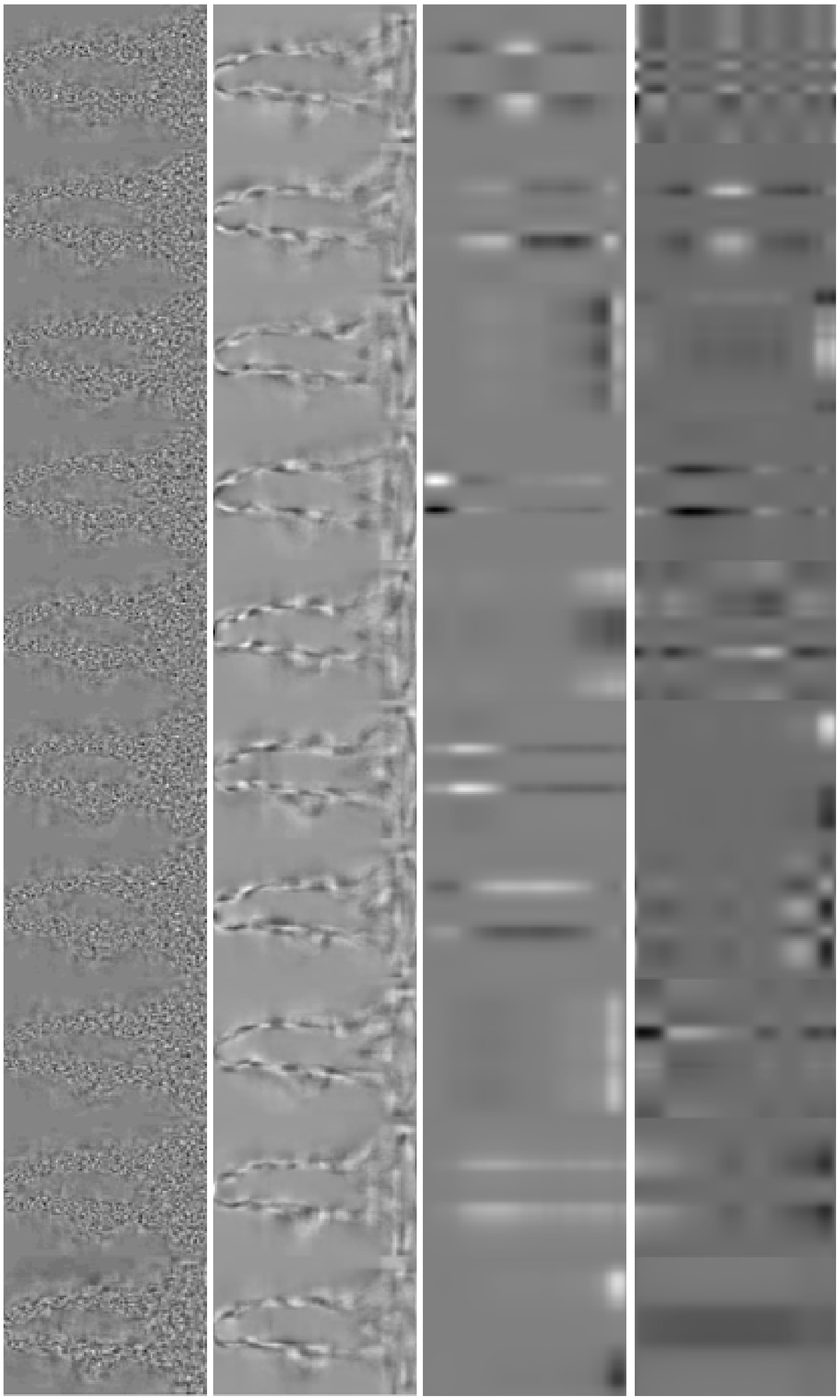


Figure 4.18. First 10 EigenGaits (the first column), first 10 FisherGaits (the second column), first 10 TR1AGaits (the third column), and first 10 TR1DAGaits (the fourth column). From the figure, we can see that EigenGaits and FisherGaits are dense, while TR1AGaits and TR1DAGaits are sparse, because they take the

structure information into account to reduce the number of unknown parameters in discriminant learning.

Figure 4.19–Figure 4.26 illustrate the experimental results under eight different circumstances. In each of them:

- The *first sub–figure* shows the effects of feature dimension on recognition precision (one minus the error rate). In this sub–figure, the x–coordinate is the feature dimension and the y–coordinate is the recognition precision. In order to keep the graph be in a manageable size, we only show the results for feature dimensions from 10 to 110. In these experiments, we show the top–one recognition precisions of EigenGait, FisherGait, TR1AGait, and TR1DAGait. We show the top–one and top–five recognition precisions of these algorithms in Table 4.5 and Table 4.6, respectively. Usually, tensor based algorithms (TR1AGait and TR1DAGait) achieve better recognition precisions than vector based algorithms (EigenGait and FisherGait) and TR1DAGait performs better than TR1AGait.

- The *second sub–figure* shows the effects of feature dimension and the tuning parameter ς on the recognition precision in TR1DA. In this sub–figure, the x–coordinate is the feature dimension; the y–coordinate is the tuning parameter ς; and the z–coordinate is the recognition precision. The feature dimension changes from 10 to 200 with a step 1 and ς changes from 0.01 to 0.5 with a step 0.01. For each probe the tuning parameter ς is chosen to achieve the best performance. The detailed information about the value of ς is given in Table 4.4. In each case there is a range of values of ς for which TR1DA achieves a good performance. We only show one value for each probe. It can be observed from the sub–figure that for each probe, there are a number of points to achieve the best performance according to different ς values.

- The *third sub–figure* shows the number of iterations required in the training procedure for extracting the i^{th} feature. In this sub–figure, the x–coordinate is the feature dimension and the y–coordinate is the number of iterations required for convergence. The mean value of the number of iterations for convergence is represented by the dashed line. In these experiments, we set the maximum number of training iterations as 1,000. The training procedure usually converges within 80 iterations. The mean number of training iterations for different features is about 20 and the standard deviation is about 10. Detailed information about the number of iterations required can be found in Table 4.4.

- The four *bottom sub–figures* examine the training procedure convergence property of TR1DA. Because different features share similar convergence curves, we only show the convergence curves of the first two features. The left column is relevant to the 1st feature and the right column is relevant to the 2nd feature. In the upper two of the four bottom–right sub–figures, the x–coordinate is the number of the training iterations *t* and the y–coordinate is the log of differences in a projection direction between two neighboring training iterations, i.e., they demonstrate how $\log\left(\left\|\left(u_k\right)_t^T\left(u_k\right)_{t-1}\right|-1\right| \Big/ \left|\left(u_k\right)_t\right|_0\right)$ and $\log\left(\left\|\left(v_k\right)_t^T\left(v_k\right)_{t-1}\right|-1\right| \Big/ \left|\left(v_k\right)_t\right|_0\right)$ (mentioned in Table 4.2) change with the training iterations for the 1st and 2nd features ($k = 1,2$). Here, $|x|_0$ is the dimension of x; $\left(u_k\right)_t$ is the first projection vector of the *t*th training iteration for the *k*th feature; and $\left(v_k\right)_t$ is the second projection vector of the *t*th training iteration for the *k*th feature. In these experiments, the visual objects are averaged gait images, which are second order tensors. To avoid confusion, we use u and v to represent the projection vectors in different directions. From the upper two sub–figures, we can see that as the number of training iterations increases, the changes in u and v approach to zero. In the lower two of the four bottom–right sub–figures, the x–coordinate denotes the number of training iterations and the y–coordinate is the function value f of TR1DA defined in (13), i.e., they show how the function value of TR1DA defined in (13) changes with the training iterations for the 1st and 2nd features. From the sub–figures, we can see that as the number of training iterations increases, the change of the function value of TR1DA approaches zero. All of these sub–figures demonstrate that the training procedure of

TR1DA converges after about 20 iterations. If $\log\left(\left|\left|(u_k)_t^T (u_k)_{t-1}\right|-1\right| \Big/ \left|(u_k)_t\right|_0\right) < \varepsilon$ and $\log\left(\left|\left|(v_k)_t^T (v_k)_{t-1}\right|-1\right| \Big/ \left|(v_k)_t\right|_0\right) < \varepsilon$, we deem the training procedure converges. The value of ε is 10^{-6}.

Table 4.4. Parameters in convergence examination for eight probes.

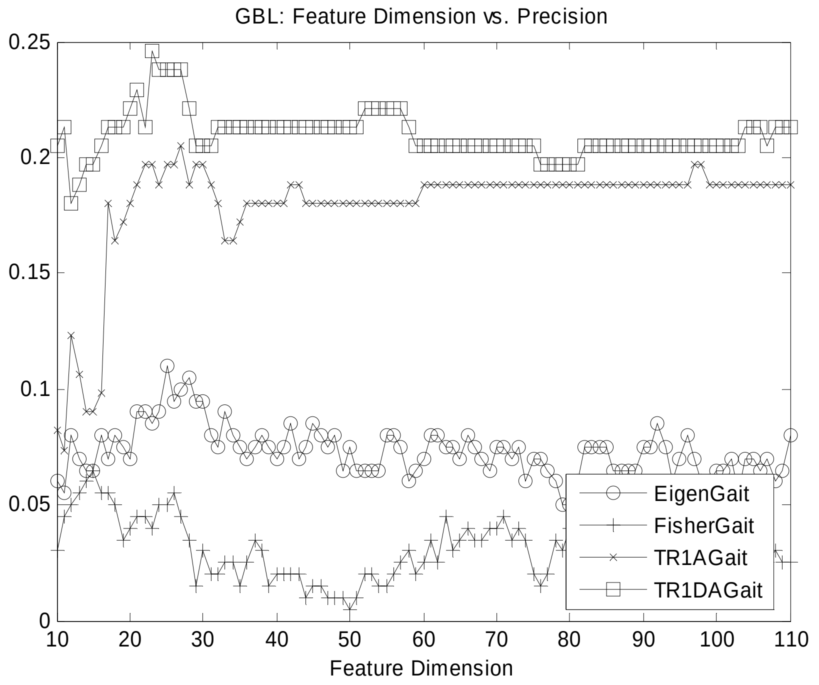
TR1DA	GBL	GAL	CBL	CAL	GBBL	GBAL	CBBL	CBAL
Max	58	63	55	67	65	73	54	53
Mean	21	19	18	19	20	22	18	19
Std	11	10	10	11	11	14	11	10
ς	0.03	0.17	0.25	0.31	0.02	0.05	0.20	0.28

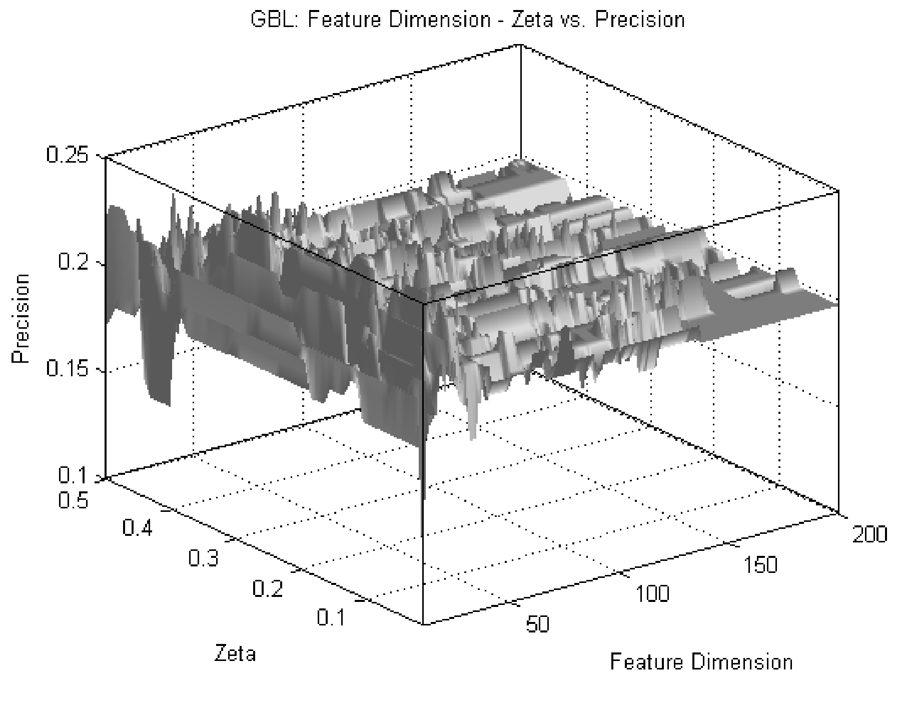
Table 4.5. Rank One recognition precision for eight probes.

Rank One	GBL	GAL	CBL	CAL	GBBL	GBAL	CBBL	CBAL
EigenGait	11.00	14.66	13.44	18.28	09.50	10.00	11.29	09.68
FisherGait	07.50	17.80	13.98	**18.82**	09.00	13.00	07.53	16.13
TR1AGait	20.49	17.80	18.11	13.98	**23.48**	15.00	17.74	15.98
TR1DAGait	**24.59**	**18.85**	**21.26**	16.13	**23.48**	**16.00**	**20.97**	**17.16**

Table 4.6. Rank five recognition precision for eight probes.

Rank Five	GBL	GAL	CBL	CAL	GBBL	GBAL	CBBL	CBAL
EigenGait	28.00	29.84	33.87	**41.94**	31.00	24.50	31.72	30.11
FisherGait	25.00	**32.98**	40.32	**41.94**	24.00	**29.00**	23.12	33.87
TR1AGait	36.07	30.89	40.94	33.33	38.64	28.50	35.48	35.50
TR1DAGait	**37.70**	30.37	**44.09**	34.41	**39.39**	**29.00**	**36.29**	**36.69**


GBL: Feature Dimension vs. Precision
0.25
0.2
0.15
0.1
0.05
0
10
20
30
40
50
60
70
80
90
100
110
Feature Dimension
EigenGait
FisherGait
TR1AGait
TR1DAGait


GBL: Feature Dimension - Zeta vs. Precision
Precision
0.25
0.2
0.15
0.1
0.5
0.4
0.3
0.2
0.1
Zeta
50
100
150
200
Feature Dimension

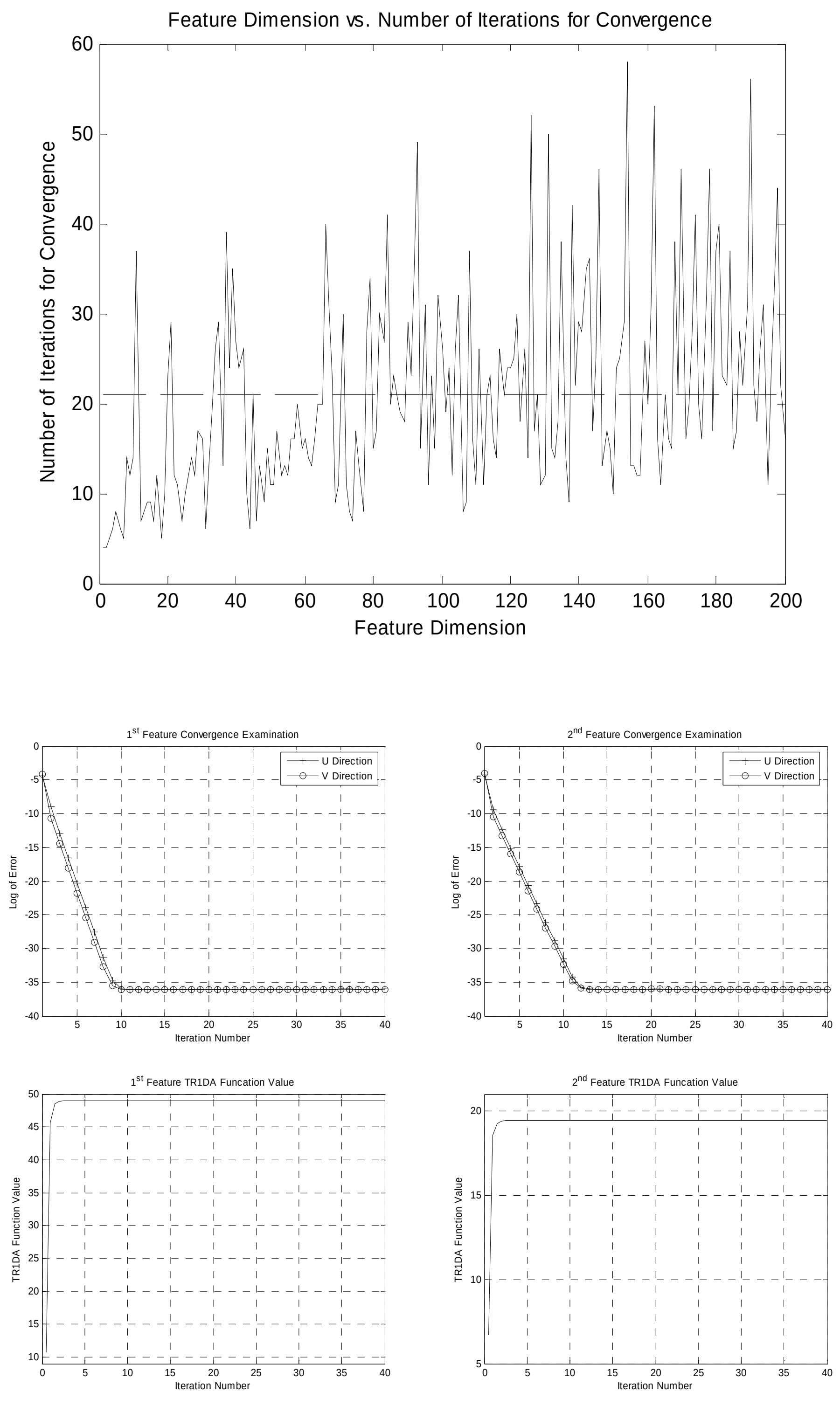
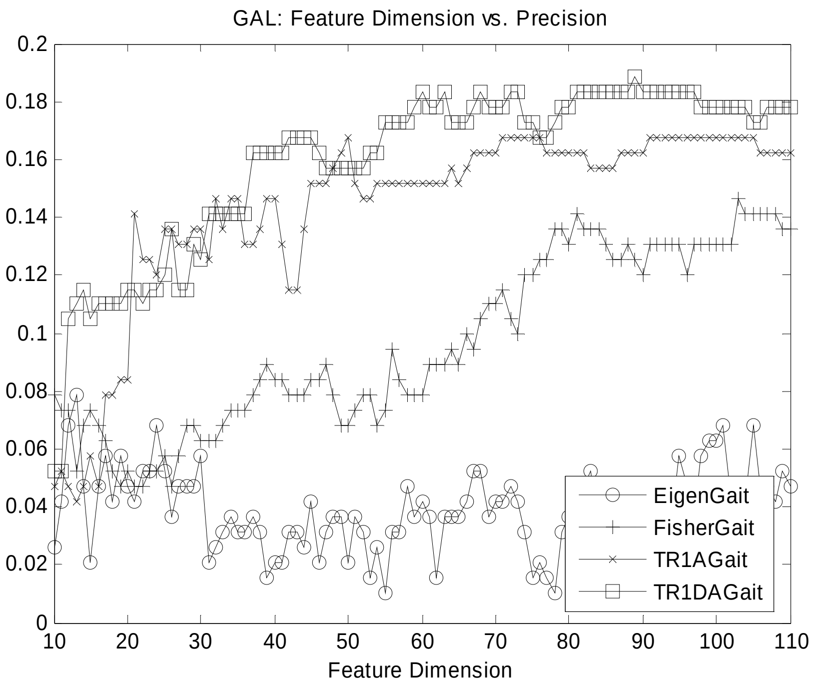
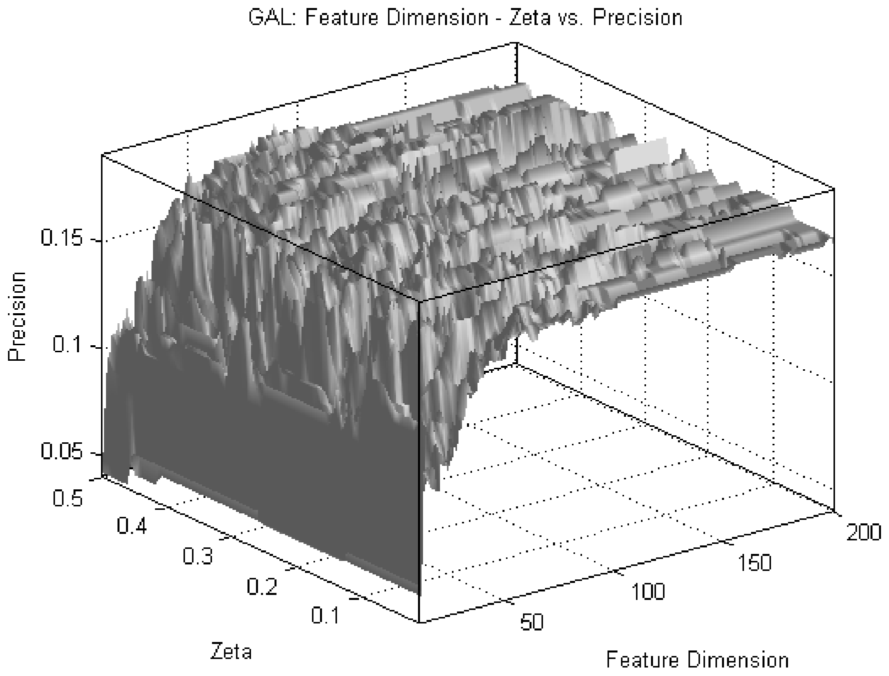
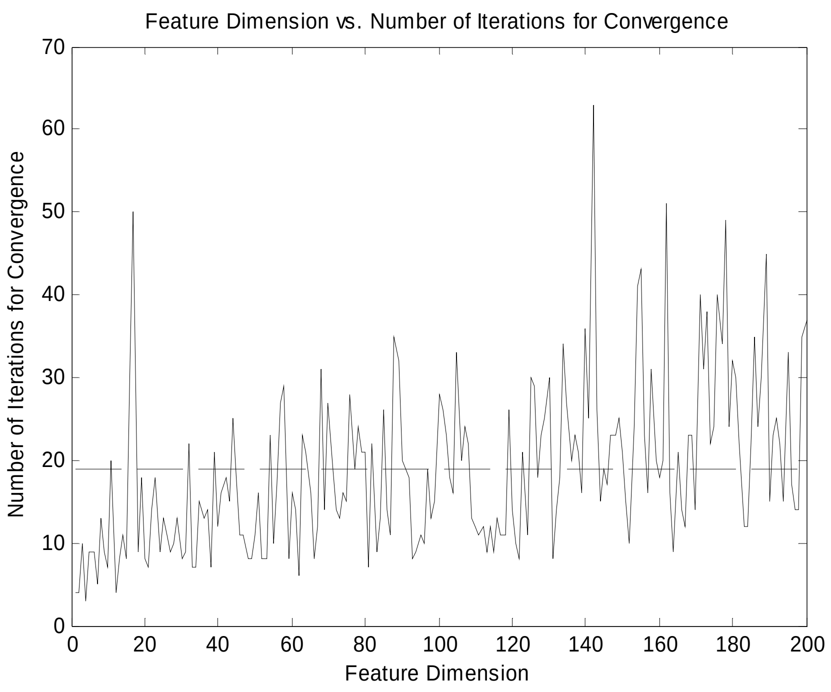


Figure 4.19. GBL task: TR1DA performances, convergence, and parameters (feature dimension and ς) analysis. Detailed explanation is in §233066540.


GAL: Feature Dimension vs. Precision
0.2
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
10
20
30
40
50
60
70
80
90
100
110
Feature Dimension
EigenGait
FisherGait
TR1AGait
TR1DAGait


GAL: Feature Dimension - Zeta vs. Precision
Precision
0.15
0.1
0.05
0.5
0.4
0.3
0.2
0.1
Zeta
50
100
150
200
Feature Dimension



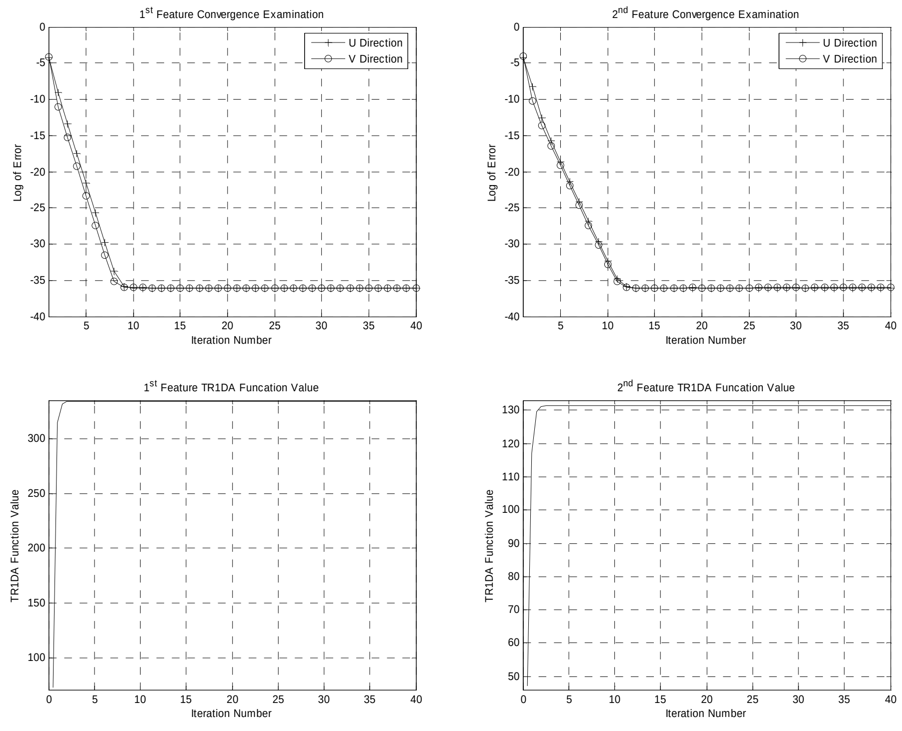
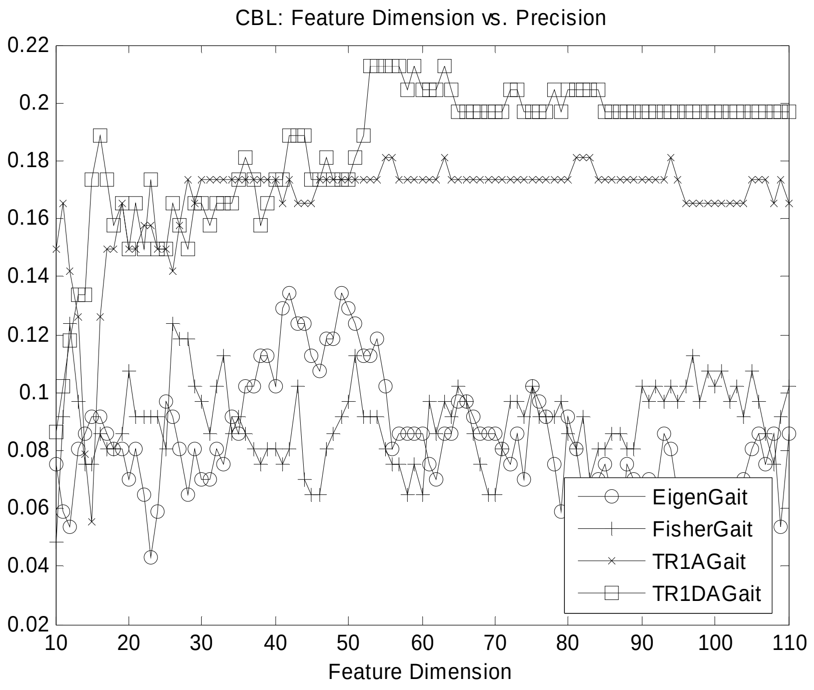
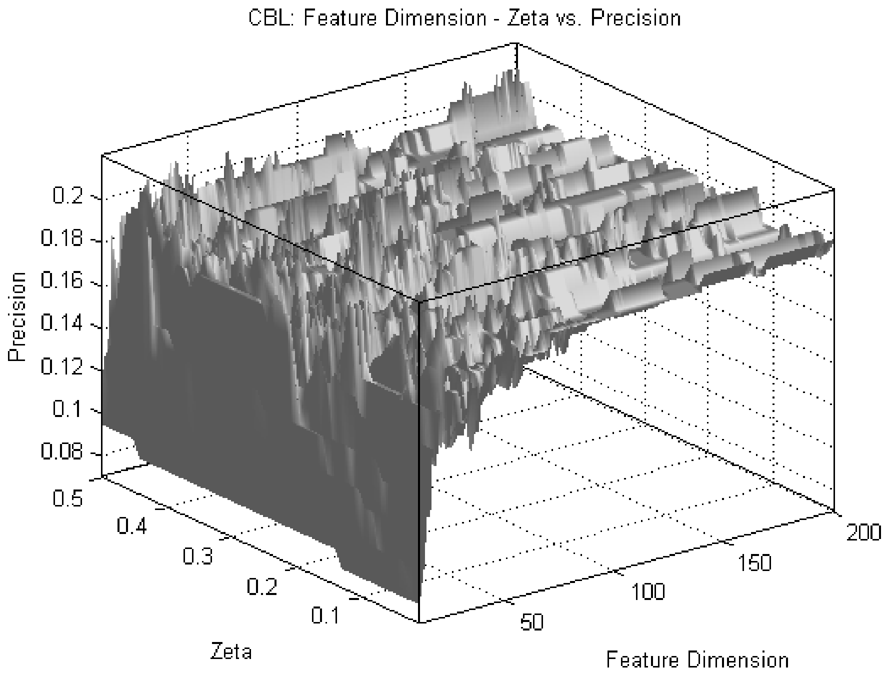
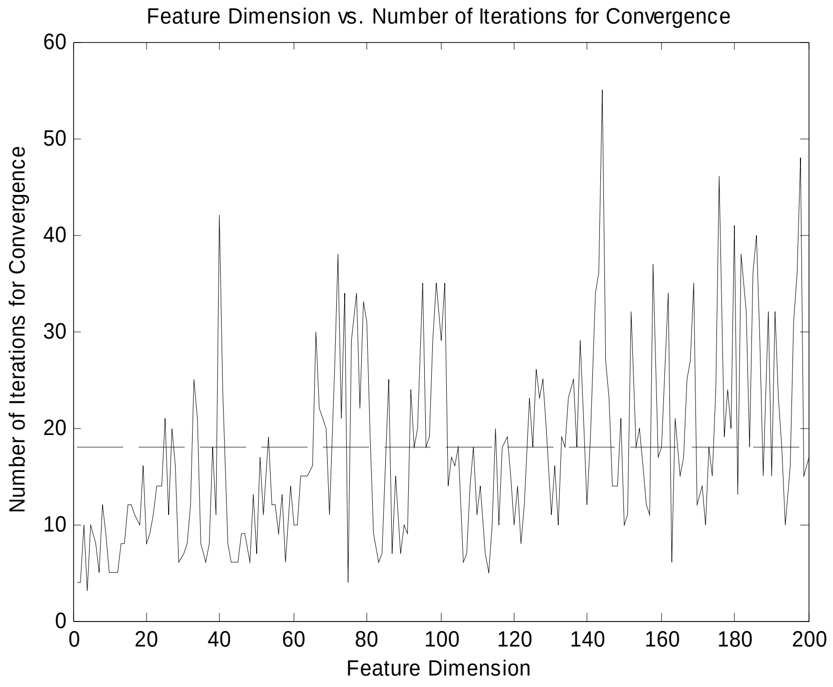


Figure 4.20. GAL task: TR1DA performances, convergence, and parameters (feature dimension and ς) analysis. Detailed explanation is in §0.


CBL: Feature Dimension vs. Precision
0.22
0.2
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
10
20
30
40
50
60
70
80
90
100
110
Feature Dimension
EigenGait
FisherGait
TR1AGait
TR1DAGait


CBL: Feature Dimension - Zeta vs. Precision
Precision
0.2
0.18
0.16
0.14
0.12
0.1
0.08
0.5
0.4
0.3
0.2
0.1
Zeta
50
100
150
200
Feature Dimension



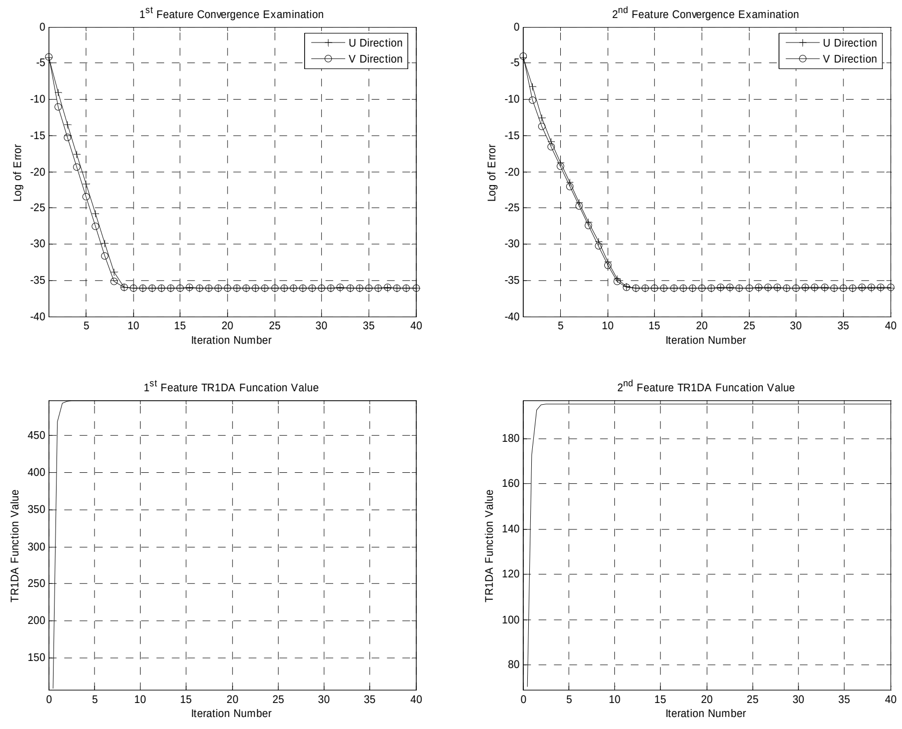
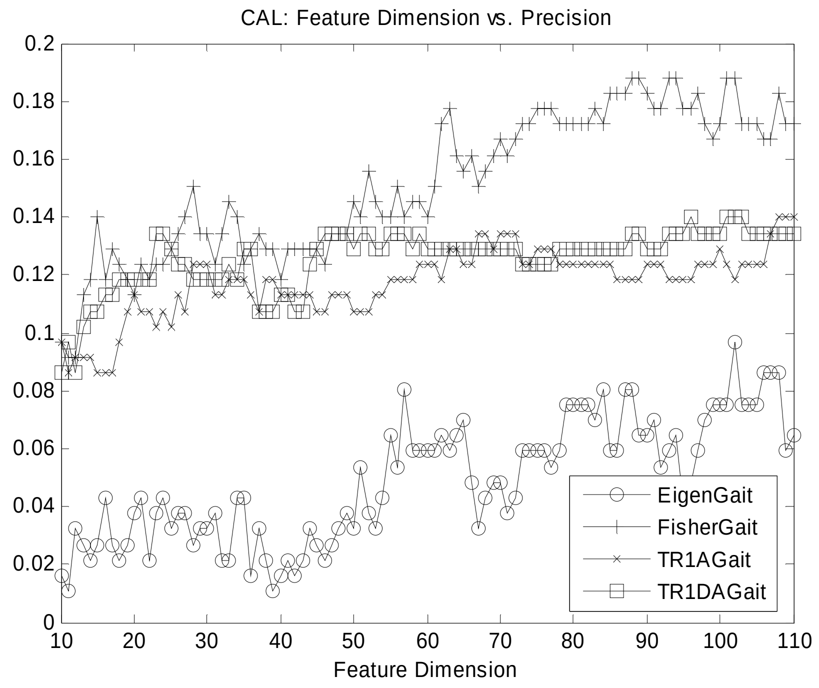
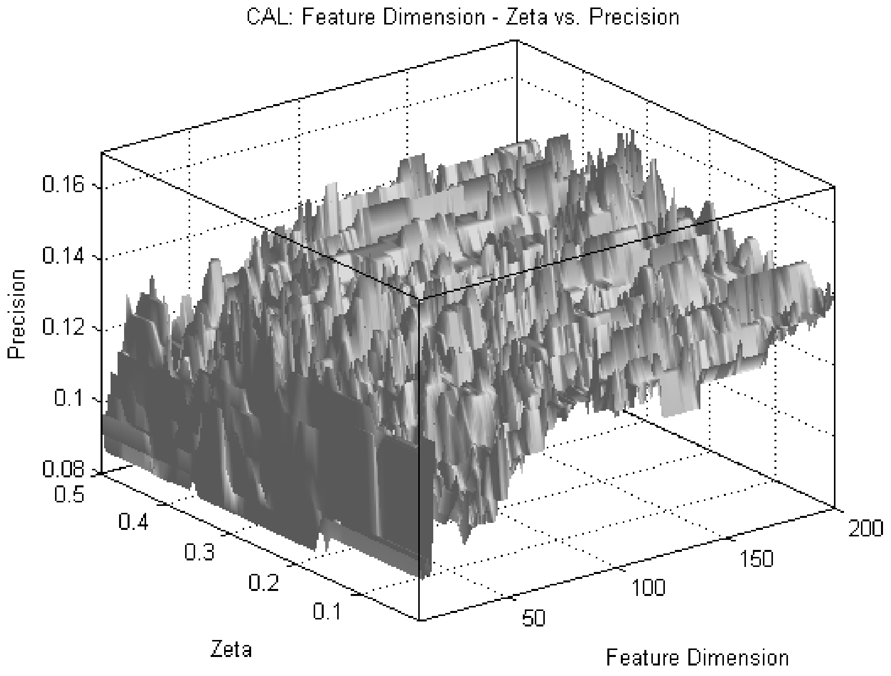


Figure 4.21. CBL task: TR1DA performances, convergence, and parameters (feature dimension and ς) analysis. Detailed explanation is in §0.


CAL: Feature Dimension vs. Precision
0.2
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
10
20
30
40
50
60
70
80
90
100
110
Feature Dimension
EigenGait
FisherGait
TR1AGait
TR1DAGait


CAL: Feature Dimension - Zeta vs. Precision
Precision
0.16
0.14
0.12
0.1
0.08
0.5
0.4
0.3
0.2
0.1
Zeta
50
100
150
200
Feature Dimension

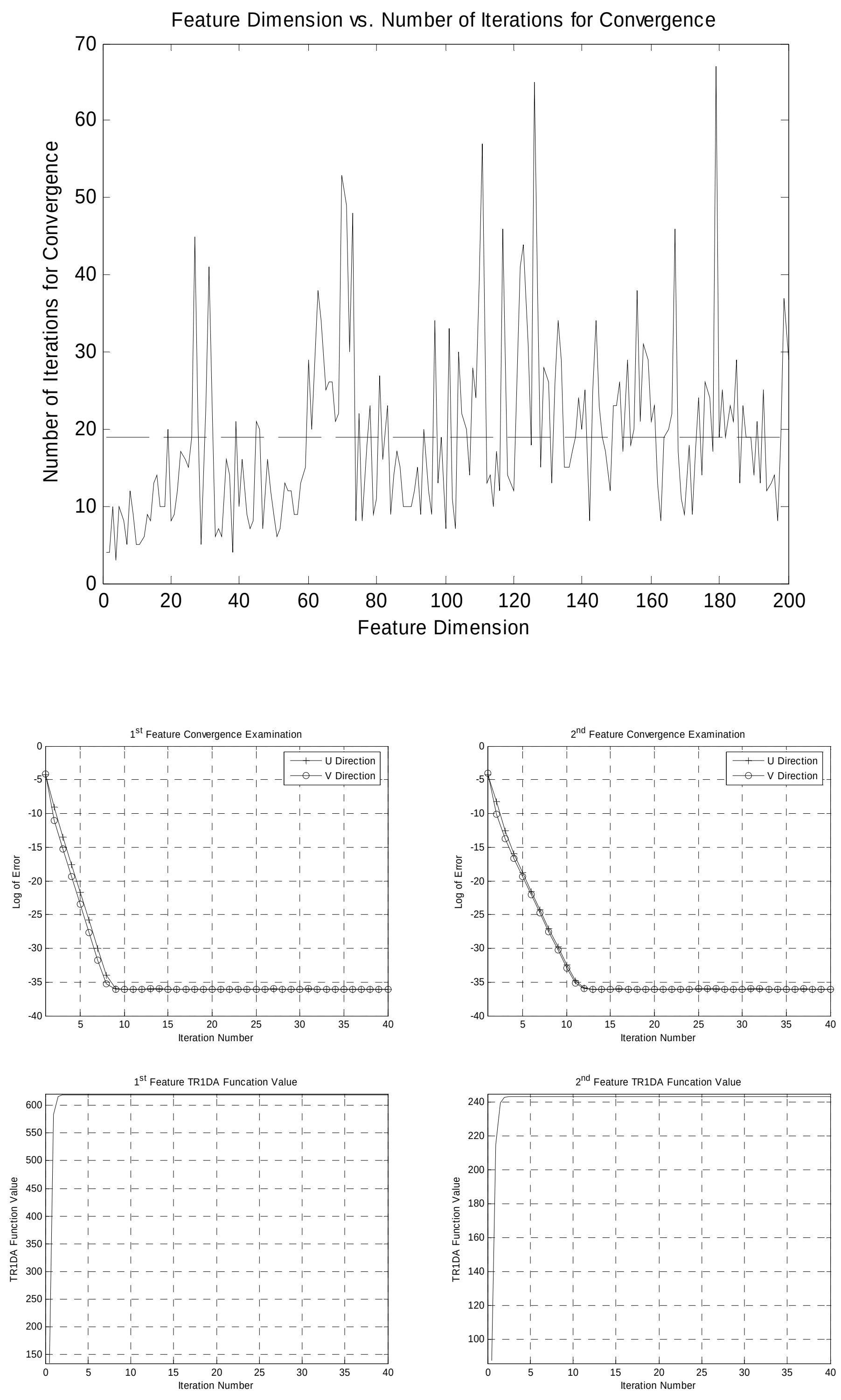
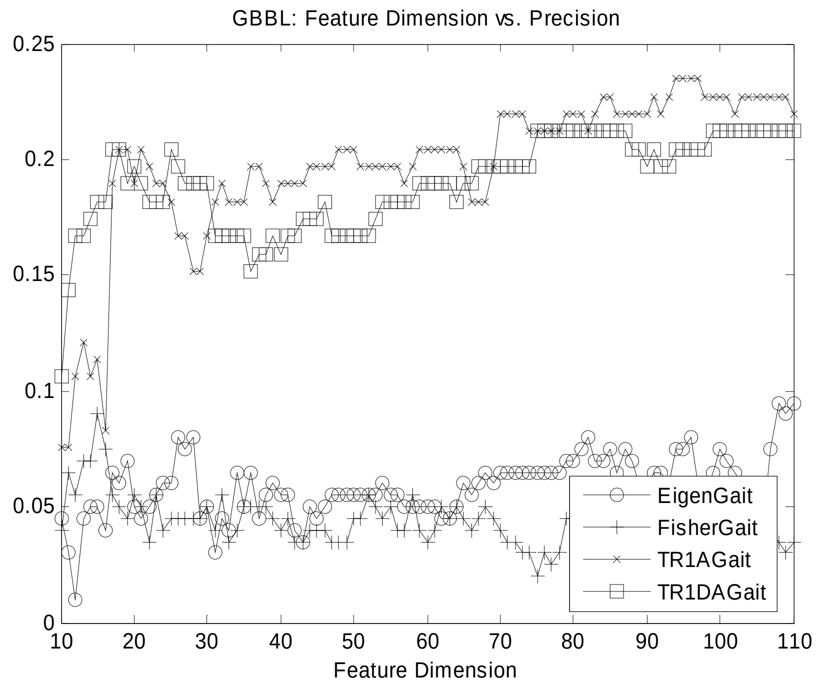
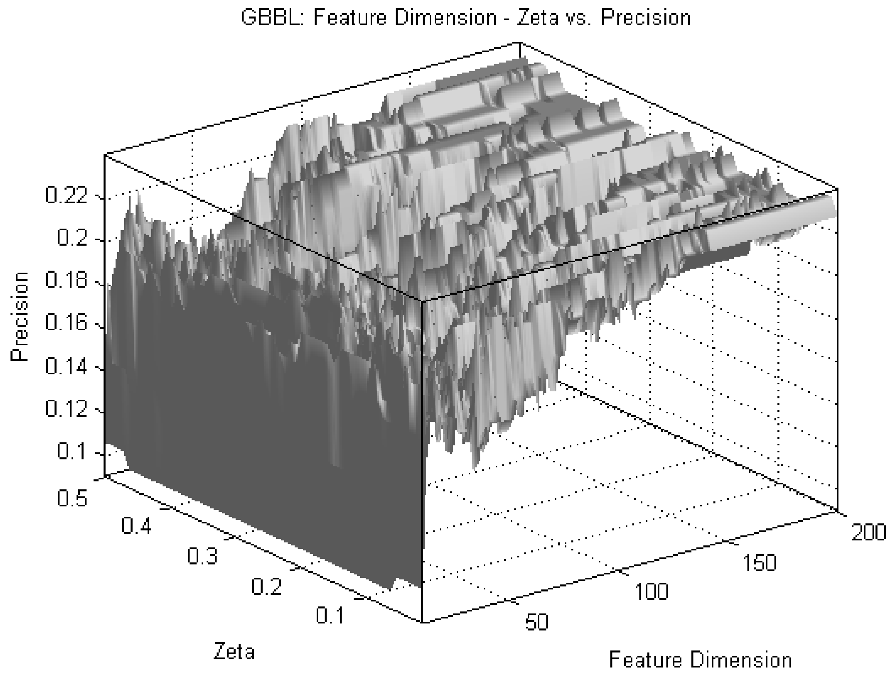


Figure 4.22. CAL task: TR1DA performances, convergence, and parameters (feature dimension and ς) analysis. Detailed explanation is in §0.


GBBL: Feature Dimension vs. Precision
0.25
0.2
0.15
0.1
0.05
0
10
20
30
40
50
60
70
80
90
100
110
Feature Dimension
EigenGait
FisherGait
TR1AGait
TR1DAGait


GBBL: Feature Dimension - Zeta vs. Precision
Precision
0.22
0.2
0.18
0.16
0.14
0.12
0.1
0.5
0.4
0.3
0.2
0.1
Zeta
50
100
150
200
Feature Dimension

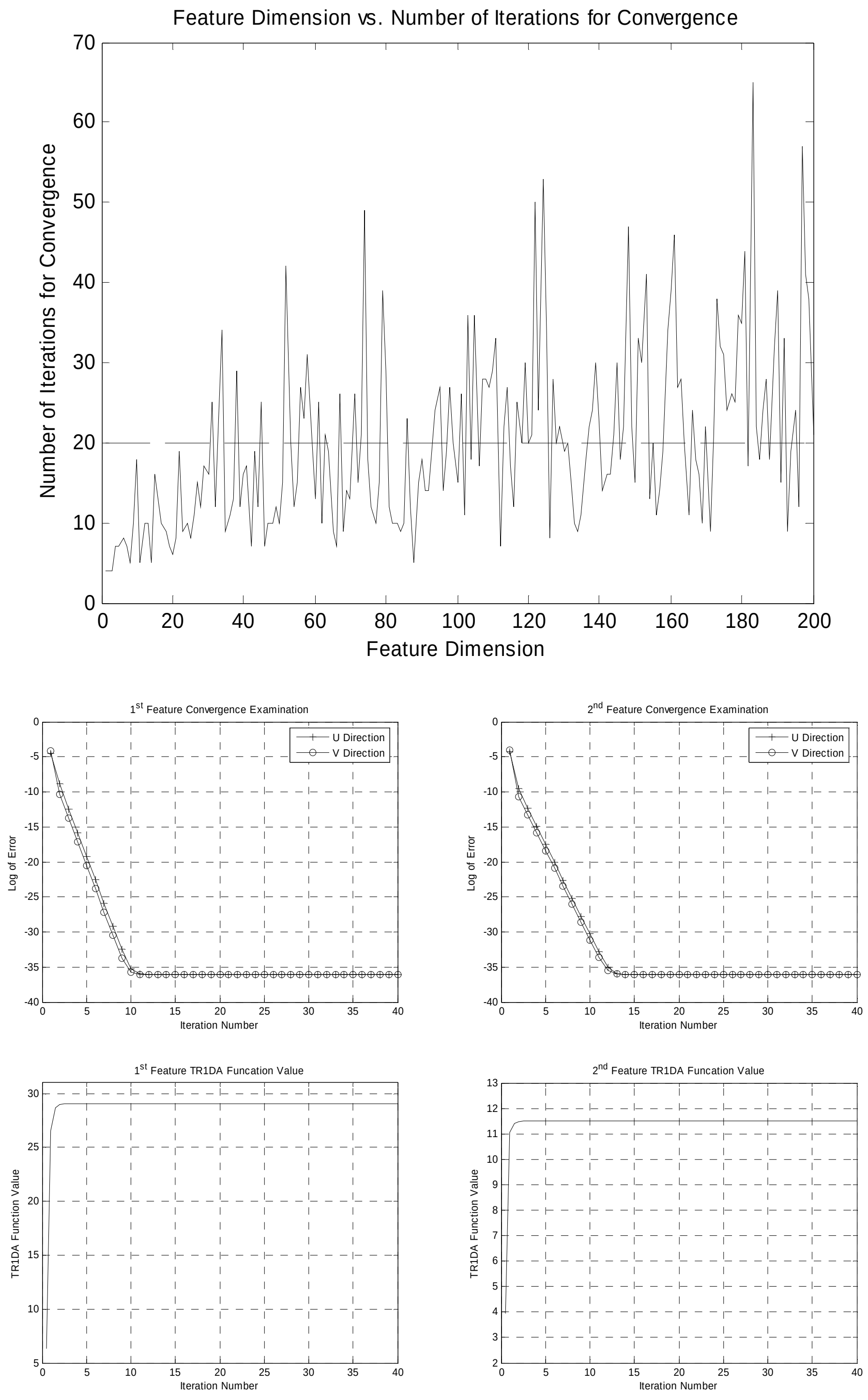
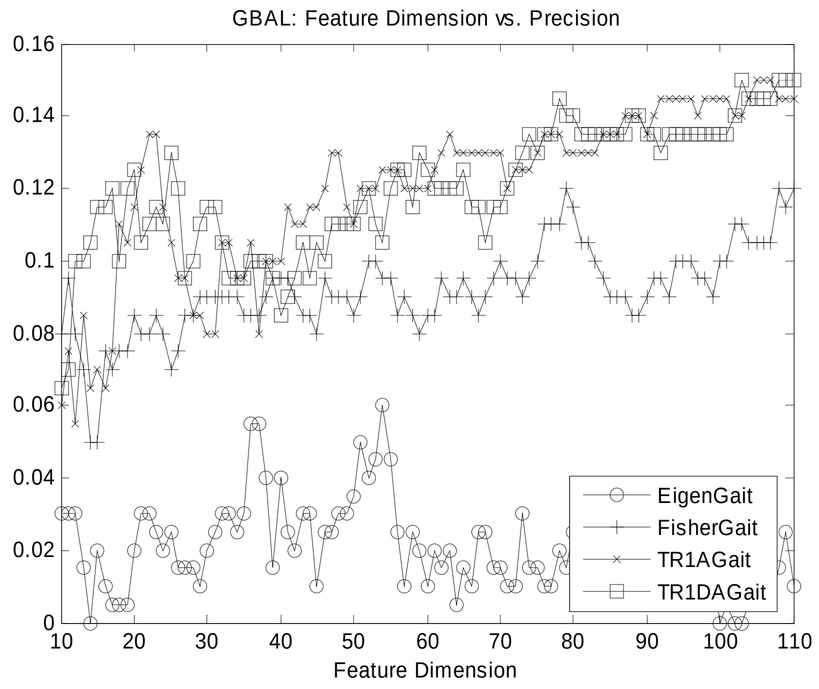
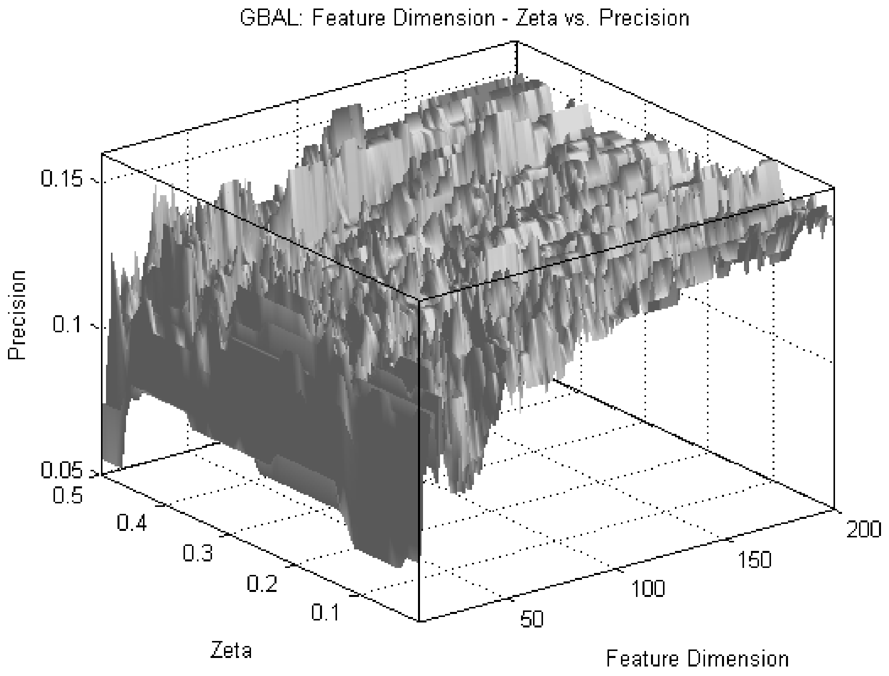


Figure 4.23. GBBL task: TR1DA performances, convergence, and parameters (feature dimension and ς) analysis. Detailed explanation is in §0.


GBAL: Feature Dimension vs. Precision
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
10
20
30
40
50
60
70
80
90
100
110
Feature Dimension
EigenGait
FisherGait
TR1AGait
TR1DAGait


GBAL: Feature Dimension - Zeta vs. Precision
0.15
0.1
0.05
Precision
0.5
0.4
0.3
0.2
0.1
Zeta
50
100
150
200
Feature Dimension

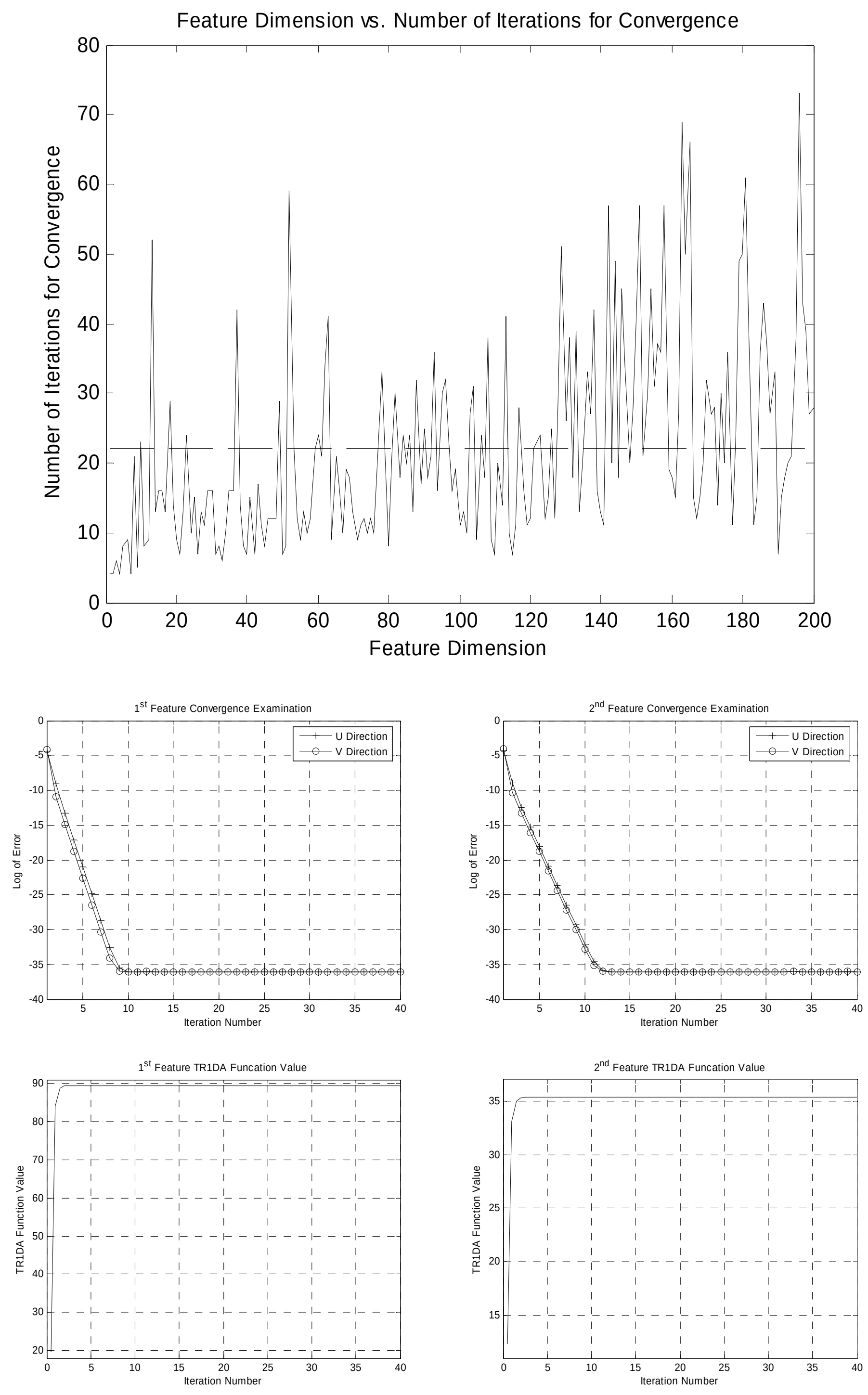
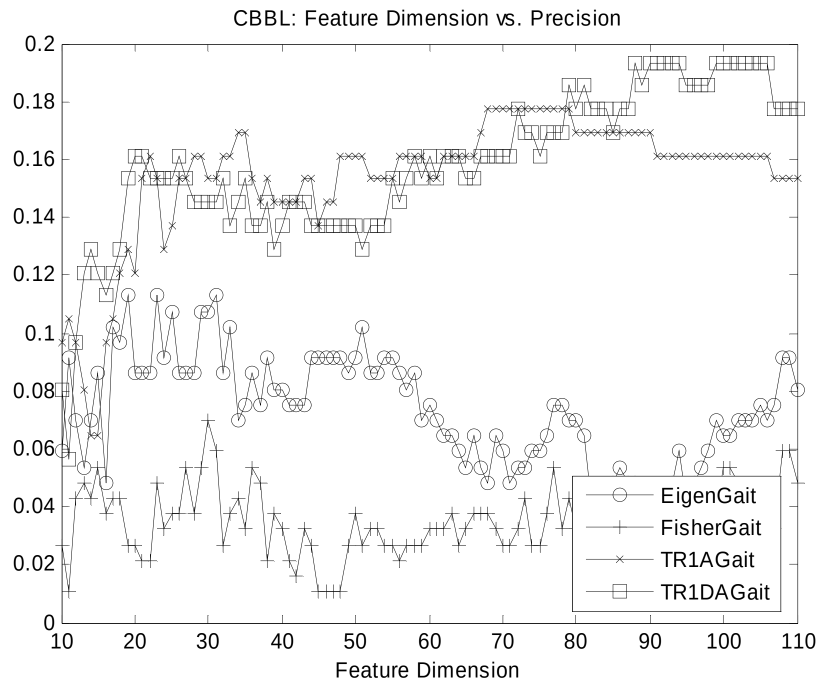
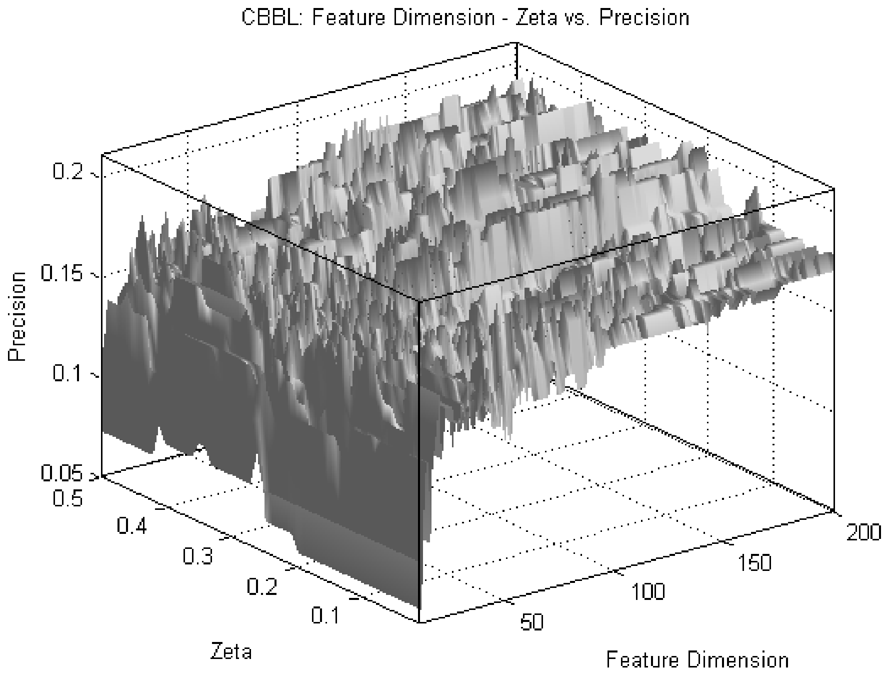


Figure 4.24. GBAL task: TR1DA performances, convergence, and parameters (feature dimension and ς) analysis. Detailed explanation is in §0.


CBBL: Feature Dimension vs. Precision
Feature Dimension
EigenGait
FisherGait
TR1AGait
TR1DAGait
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2
10
20
30
40
50
60
70
80
90
100
110


CBBL: Feature Dimension - Zeta vs. Precision
Precision
Zeta
Feature Dimension
0.05
0.1
0.15
0.2
0.5
0.4
0.3
0.2
0.1
50
100
150
200

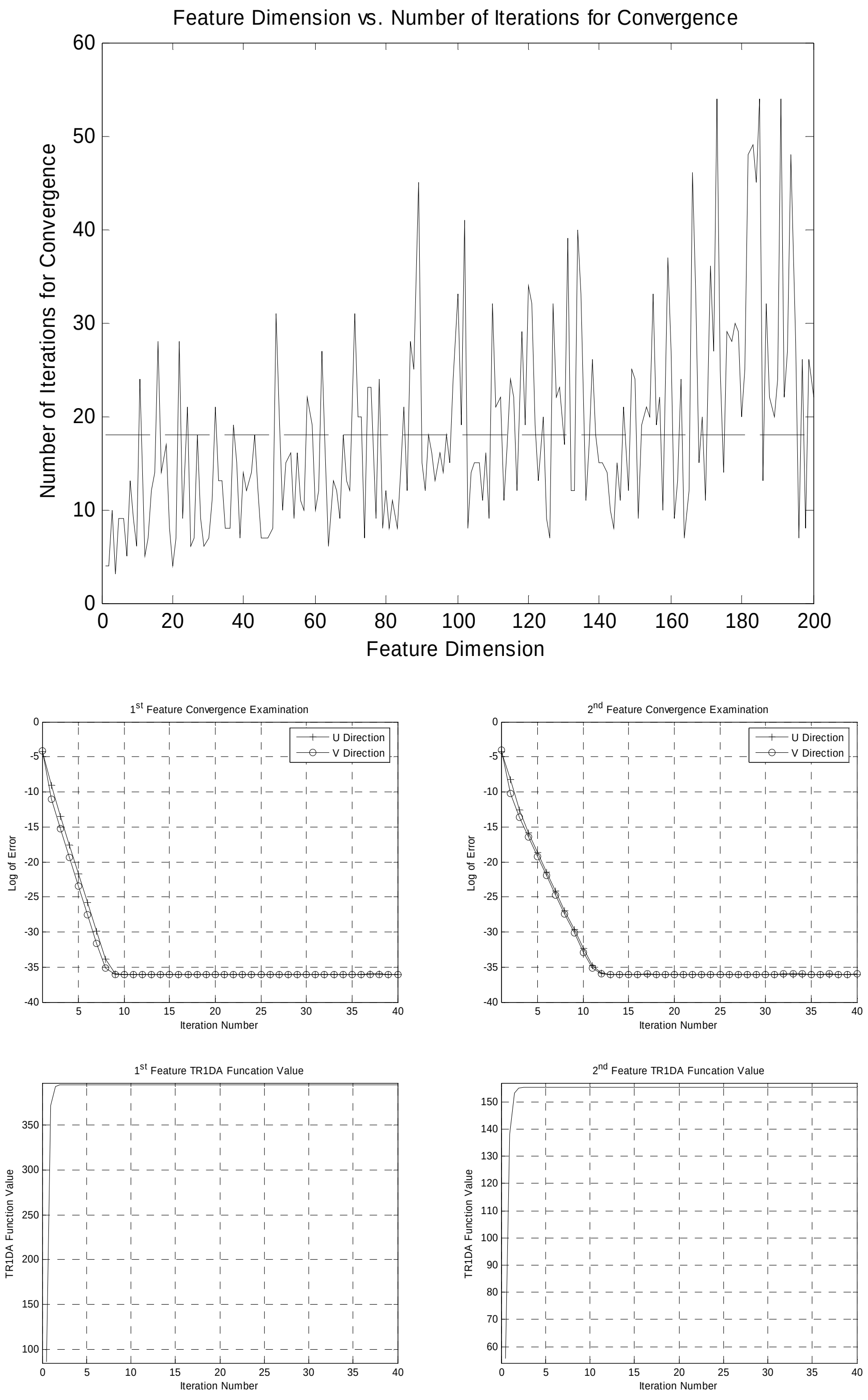
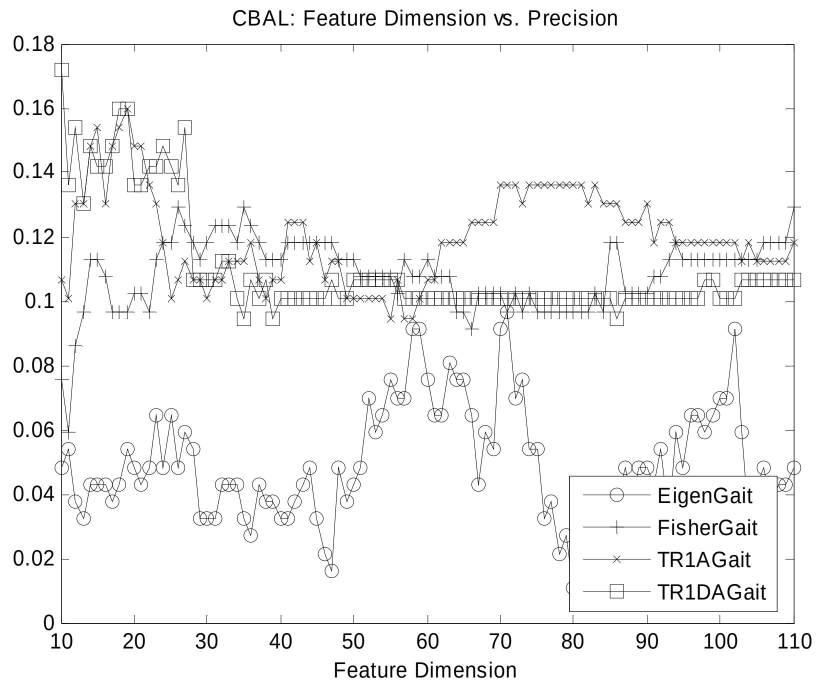
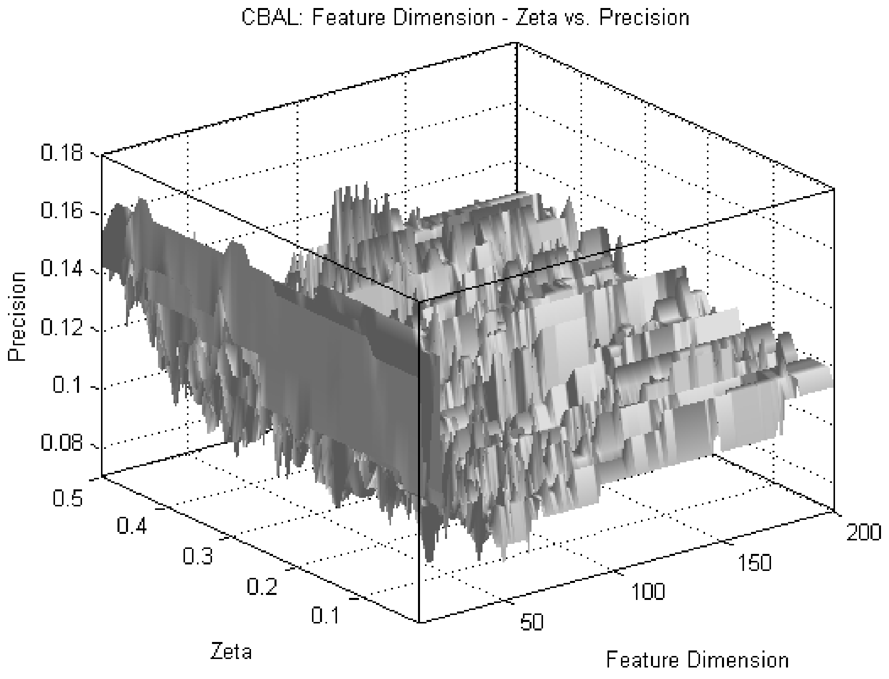


Figure 4.25. CBBL task: TR1DA performances, convergence, and parameters (feature dimension and ς) analysis. Detailed explanation is in §0.


CBAL: Feature Dimension vs. Precision
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
10
20
30
40
50
60
70
80
90
100
110
Feature Dimension
EigenGait
FisherGait
TR1AGait
TR1DAGait


CBAL: Feature Dimension - Zeta vs. Precision
Precision
0.08
0.1
0.12
0.14
0.16
0.18
0.5
0.4
0.3
0.2
0.1
Zeta
50
100
150
200
Feature Dimension

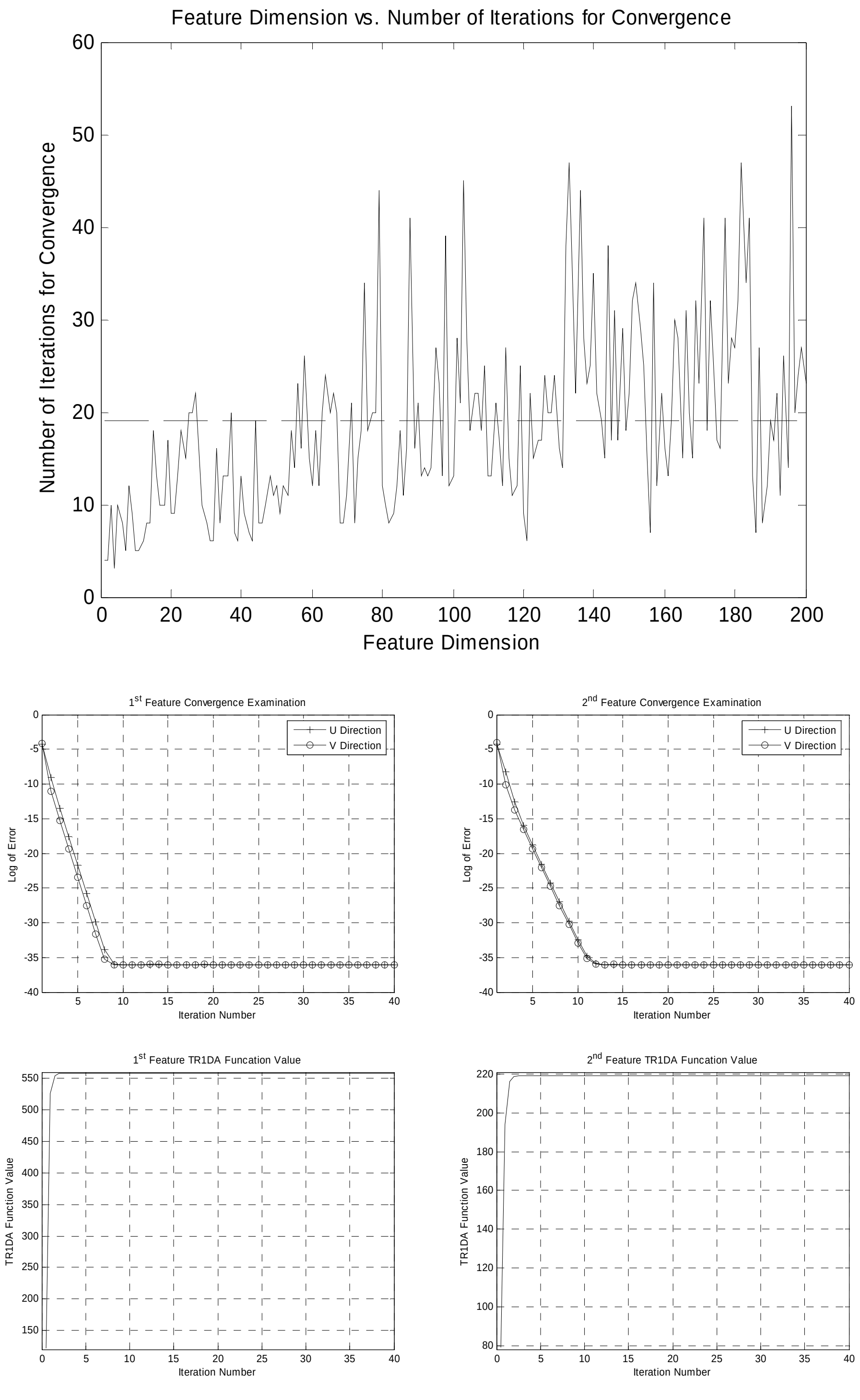


Figure 4.26. CBAL task: TR1DA performances, convergence, and parameters (feature dimension and ς) analysis. Detailed explanation is in §0.

Based on Figures 4.19 – 4.26, we have the following observations:

1. The first sub–figure of each figure shows the classification performance versus the selected feature dimension. Most of these sub-figures show that

tensor based subspace selection methods (TR1DA and TR1A) perform better than vector based ones (LDA and PCA) and the discriminative subspace selection methods (TR1DA and LDA) perform better than reconstructive subspace selection methods (TR1A and PCA). This is because the size of the gallery, i.e., the training set, in different tasks is much less than the dimension of original features, i.e., the appearance of the averaged gait image for representation. According to the discussions at the beginning of this Chapter, we know tensor based subspace selection methods reduce the over fitting problem when training samples are limited, so TR1DA and TR1A should perform better than LDA and PCA, respectively. Moreover, tasks here are classification so discriminative based methods perform better than reconstructive based methods as discussed at the beginning of Chapter 2;

2. The second sub–figure of each figure shows the classification performance of TR1DA versus the parameters in TR1DA. There are two parameters for TR1DA, which are ς and selected feature dimension. Some of these sub-figures show that the more the selected features are, the higher the classification accuracy is for the first selected 200 features; and some of them show that with the increasing number of selected features, the classification accuracy is initially increasing and then decreasing. From a large number of experiences in subspace selection, we know the latter is more reasonable than the former, so we believe we can have the similar observations as the latter if more features are selected for evaluation;
3. The third sub–figure of each figure shows the number of iterations required to meet the training stage convergence condition in TR1DA for different features. According to Table 4.4, normally, TR1DA needs about 20 iterations to converge and the corresponding standard deviation is about 10. Therefore, for most features, we need about 30 iterations for training. The distributions of the number of training iterations are similar to noise, i.e., there is no clear relationship between the number of training iterations and the selected features. This is because the training processes for different features are independent, as shown in Section 4.4; and
4. The last four sub–figures of each figure demonstrate the convergence property of the training stage of TR1DA. The first two sub–figures show that the difference between the projection vectors of two consecutive training

iterations is decreasing with the increasing number of the training iterations; and the last two sub–figures show the TR1DA function value is increasing with the increasing number of the training iterations. These two points consist with the mathematical proof of the theorem 4.1.

Summary

In this Chapter, the vector based learning is extended to accept tensors as input. The result is a supervised tensor learning (STL) framework, which is the multilinear extension of convex optimization based learning. To obtain the solution of an STL based learning algorithm, the alternating projection method is used. Based on STL and its alternating projection optimization algorithm, we illustrate several examples. That is we extend the soft margin support vector machine (SVM), the nu–SVM, the least squares SVM, the minimax probability machine (MPM), the Fisher discriminant analysis (FDA), the distance metric learning (DML) to their tensor versions, which are the soft margin support tensor machine (STM), the nu–STM, the least squares STM, the tensor MPM (TMPM), the tensor FDA (TFDA), and the multiple distance metrices learning (MDML).

Based on STL, we also introduce a method for iterative feature extraction: the iterative feature extraction model (IFEM). As an example, we develop the tensor rank one discriminant analysis (TR1DA).

Finally, we implement TMPM for image classification and TR1DA for the elapsed time problem in human gait recognition. By comparing TMPM with MPM, we show that TMPM reduces the overfitting problem in MPM. By comparing TR1DA with principal component analysis (PCA), linear discriminant analysis (LDA), and tensor rank one analysis (TR1A), we show that TR1DA reduces the small sample size problem and always achieves the best performance on the elapsed time problem in human gait recognition.

5. Thesis Conclusion

Linear discriminant analysis (LDA) motivates this thesis. This is because LDA has many problems, although it has been deemed as one of the most important linear subspace methods in pattern classification and it has been widely applied in many applications, e.g., face recognition [34][168], image retrieval [139][152] [153][154], video data organization [43], gait recognition [46], speech recognition [67][70], document classification [108], music management [85], network flow analysis [136], and video surveillance [143][144].

The first type of the problems with LDA is model based: 1) the heteroscedastic problem [29][28][70][61][99], 2) the multimodal problem [51][28], and 3) the class separation problem [103][95][96][99][146]. To deal with the model based problems, we begin with the fact that LDA selects the subspace to maximize the arithmetic mean of the Kullback–Leibler (KL) [21] divergences between different classes, when samples are sampled from Gaussian distributions [60] with identical covariances. We then generalize LDA in two ways: 1) extending the KL divergence to the Bregman divergence [14]; and 2) extending the arithmetic mean to the generalized mean [48]. The result of these generalizations is the general averaged divergences analysis and the significance of the generalization is a discriminative subspace selection framework, from which we can develope a number of different methods to select discriminative features. Based on this framework, we analyze the geometric based subspace selection: 1) the geometric mean of divergences between different classes, 2) the geometric mean of normalized divergences between different classes, and 3) the geometric mean of all divergences (the divergences and the normalized divergences) between different classes. The first method is studied because the geometric mean increases the effects of small divergences and at the same time reduces the effects of large divergences. The second method is studied to further reduce the effects of large divergences. The intuition is the product of normalized divergences is large when they are similar to each other. However, the second method cannot be directly used for subspace selection. This is because there are subspaces in which all divergences become small, but all normalized divergences are comparable in size. Consequently, we linearly combine the first and the second methods together. To apply the geometric mean based subspace selection for real applications, we suppose the samples are sampled from Gaussian distributions and we use the KL divergence to measure differences between different classes. Because we drop the identical covariances assumption in LDA, we do not meet the heteroscedastic problem. The advantages of the combination of the geometric mean and the KL divergences are: 1) it is compatible with the heteroscedastic property of the distributions of samples in different classes; 2) it selects suitable discriminative subspace when samples are drawn from Gaussian mixture models; and 3) it significantly reduces the class separation problem when KL divergences of difffernt classes are not evenly distributed. For real applications, we use the Gaussian mixture model to model the samples in each class, and thus avoid the multimodal problem. Based on a large number of experiments, from synthetic data to hand writing digital recognition, the geometric mean combined with the KL divergence outperforms LDA and its representative extensions.

The second type of these problems with LDA is the small sample size (SSS) problem [38][49][139][123][19][175][55][173][174][180]. LDA meets this problem when the number of training samples is less than the dimension of the feature space. In computer vision research, this problem can be reduced in a natural way by introducing the structure information as constraints, because objects in computer vision research are always represented by multidimensional arrays, i.e., tensors [75]. For example, a face image in face recognition, an averaged gait image in human gait recognition, and a video shot in video management. Although there are some algorithms applying the structure information for subspace selection, e.g., tensor rank one analysis (TR1A) [132], general tensor analysis (GTA) [75][162][171], and two dimensional linear discriminant analysis (2DLDA) [174], each of these methods has its own drawbacks for classification. TR1A and GTA are reconstructive models, i.e., they are used to produce representations for sufficient reconstruction but not for classification. The 2DLDA fails to converge in the training stage, although its effectiveness and efficiency have been demonstrated through face recognition applications. Here, we propose a different discriminative multilinear subspace method, the general tensor discriminant analysis (GTDA) [144][147], by combining the differential scatter discriminant criterion and the operations in multilinear algebra [115][75]. Compared with TR1A, GTA, and 2DLDA, GTDA has the following benefits: 1) provision with a converged alternating projection training algorithm to obtain a solution, while 2DLDA does not; 2) preservation of more discriminative information of training samples; 3) acceptance of general tensors as input; and 4) reduction of the SSS problem in the subsequent classification, e.g., by LDA. We further extend GTDA by combining the manifold learning [13] and the operations in multilinear algebra to make the manifold learning algorithms accept general tensors as input. To examine the effictiveness and the efficiency of GTDA, we apply it for human gait recognition. Based on a great deal of comparison, GTDA combined with LDA always achieves the top level performance for USF HumanID gait database [126].

In supervised learning [30][39], when the size of training samples is small, learning machines encounter the overfitting problem. Similar to the motivation in GTDA, we also utilize the structure information as constraints to reduce the overfitting problem by decreasing the number of parameters needed to model the training samples. This results in a supervised tensor learning (STL) [149][150] framework. The framework extends the convex optimization [11] based learning to accept general tensors as input. To obtain the solution of the algorithms under the STL framework, we develop an alternating projection method. Based on STL and its alternating projection optimization algorithm, we illustrate the following examples, which are the support tensor machine (STM), the tensor minimax probability machine (TMPM), the tensor Fisher discriminant analysis (TFDA), the multiple distance metrics learning (MDML). Motivated by TR1A and based on STL, we develop the tensor rank one discriminant analysis (TR1DA), which is an iterative discriminative feature extraction method. Finally, we implement TMPM and TR1DA for image classification and the elapsed time problem in human gait recognition, respectively. By comparing TMPM with MPM, we know TMPM reduces the overfitting problem in supervised learning. By comparing TR1DA with principal component analysis (PCA), linear discriminant analysis (LDA), and tensor rank one analysis (TR1A), we know TR1DA reduces the SSS problem.

In summary, this thesis deals with non trivial problems in discriminative subspace selection, which are how to select the most discriminative subspace for classification and how to deal with the SSS problem or over-fitting problem. The primary contributions of the thesis are as follows:

1) we develop a general averaged divergences analysis framework, which is a combination of the generalized mean function and the Bregman divergence, for discriminative subspace selection;
2) under this framework, a new method, which combines the geometric mean and the Kullback–Leibler divergence, is developed to significantly reduce the class separation problem. With theoretical analysis, we know the method also reduces the heteroscedastic problem and the multimodal problem;
3) the kernelization of this new method is also developed for nonlinear problems. Unlike existing kernel algorithms, we prove the kernel version is equivalent to the linear version followed by the kernel principal component analysis;
4) a large number of empirical studies based on both synthetic and real data show the new discriminative subspace selection method performs better than LDA and its representative extensions;
5) we propose GTDA to reduce the SSS problem for LDA. The advantages of GTDA compared with existing pre-processing methods, e.g., principal component analysis (PCA), tensor rank one analysis (TR1A), general tensor analysis (GTA), and two dimensional LDA (2DLDA), are: i) reduction of the SSS problem for subsequent classification, e.g., by LDA, ii) preservation of discriminative information in training tensors, while PCA, TR1A, and MSA do not guarantee this, iii) provision with stable recognition rates because the optimization algorithm of GTDA converges, while that of 2DLDA does not, and iv) acceptance of general tensors as input, while 2DLDA only accepts matrices as input;
6) we provide a mathematical proof to demonstrate the convergence property at the training stage. The significance of the proof is it is the first theoretical study for the convergence issue of the alternating projection based training algorithms for tensor subspace selection;
7) we apply the developed GTDA to human gait recognition. By applying Gabor filters for averaged gait image representation, GTDA for subspace selection, and LDA for classification, we achieve the stat-of-the-art performance; and

8) we develop a STL framework and an alternating projection optimization algorithm to reduce the over–fitting problem in convex optimization based learning. Based on them, we propose STM, TMPM, TFDA, and MDML. Compared with existing popular classifiers, e.g., SVM, MPM, FDA, and DML, the advantages of their tensor extensions are: i) generalizing better for the small size training set than vector based classifiers, ii) converging well in the training stage compared with existing tensor classifiers, e.g., 2DLDA, and iii) reducing training computational complexities of vector based classifiers.

6. Appendices

Appendix for Chapter 2

Let $\vec{x}_i \in R^L$ ($1 \le i \le n$) be a zero mean set of training samples, i.e., $\vec{m} = (1/n)\sum_{i=1}^{n} \vec{x}_i = 0$. Let $S = (1/n)\sum_{i=1}^{n} \vec{x}_i \vec{x}_i^T$ be the covariance matrix, and let U be the linear transformation matrix in PCA. The Frobenius norm is denoted by $\|\cdot\|_{Fro}$. We have the following properties of PCA as defined in Section 2.1.

Property 2.1: PCA maximizes the variance in the projected subspace, i.e.,

$$\arg\max_U \operatorname{tr}\left(U^T S U\right) = \arg\max_U \frac{1}{n}\sum_{i=1}^{n} \left\|U^T \vec{x}_i\right\|_{Fro}^2 .$$

Proof.

$$\arg\max_U \operatorname{tr}\left(U^T S U\right) = \arg\max_U \operatorname{tr}\left(U^T \left(\frac{1}{n}\sum_{i=1}^{n} \vec{x}_i \vec{x}_i^T\right) U\right)$$

$$= \arg\max_U \frac{1}{n}\sum_{i=1}^{n} \operatorname{tr}\left(U^T \vec{x}_i \vec{x}_i^T U\right) = \arg\max_U \frac{1}{n}\sum_{i=1}^{n} \left\|U^T \vec{x}_i\right\|_{Fro}^2$$

Therefore, PCA maximizes the variance in the projected subspace. ■

Property 2.3: PCA minimizes the reconstruction error, i.e.,

$$\arg\max_U \operatorname{tr}\left(U^T S U\right) = \arg\min_U \frac{1}{n}\sum_{i=1}^{n} \left\|\vec{x}_i - UU^T \vec{x}_i\right\|_{Fro}^2 .$$

Proof.

It is sufficient to prove $\arg\max_U \frac{1}{n}\sum_{i=1}^{n} \left\|U^T \vec{x}_i\right\|_{Fro}^2 = \arg\min_U \frac{1}{n}\sum_{i=1}^{n} \left\|\vec{x}_i - UU^T \vec{x}_i\right\|_{Fro}^2$.

$$\arg\min_U \frac{1}{n}\sum_{i=1}^{n} \left\|\vec{x}_i - UU^T \vec{x}_i\right\|_{Fro}^2$$

$$= \arg\min_U \frac{1}{n}\sum_{i=1}^{n} \left(\vec{x}_i - UU^T \vec{x}_i\right)^T \left(\vec{x}_i - UU^T \vec{x}_i\right)$$

$$= \arg\min_U \frac{1}{n}\sum_{i=1}^{n} \left(\vec{x}_i^T \vec{x}_i - \vec{x}_i^T UU^T \vec{x}_i - \vec{x}_i^T UU^T \vec{x}_i + \vec{x}_i^T UU^T UU^T \vec{x}_i\right)$$

In PCA, because $U^T U = I$, we have

$$\arg\min_U \frac{1}{n}\sum_{i=1}^{n}\left(\vec{x}_i^T\vec{x}_i - \vec{x}_i^T UU^T\vec{x}_i - \vec{x}_i^T UU^T\vec{x}_i + \vec{x}_i^T UU^T UU^T\vec{x}_i\right)$$
$$= \arg\min_U \frac{1}{n}\sum_{i=1}^{n}\left(\vec{x}_i^T\vec{x}_i - \vec{x}_i^T UU^T\vec{x}_i\right)$$
$$= \arg\min_U -\frac{1}{n}\sum_{i=1}^{n}\left(\vec{x}_i^T UU^T\vec{x}_i\right)$$
$$= \arg\max_U \frac{1}{n}\sum_{i=1}^{n}\left\|U^T\vec{x}_i\right\|_{Fro}^2$$

Therefore, PCA minimizes the reconstruction error. ■

Property 2.4: PCA decorrelates the training samples in the projected subspace.

Proof.

Let U be the projection, which is calculated according to PCA. Project all training samples $\vec{x}_i$ using U as $\vec{y}_i = U^T\vec{x}_i$, where $U \in R^{L\times L'}$.

Therefore, the covariance of the projected data is

$$S_y = \frac{1}{n}\sum_{i=1}^{n}\vec{y}_i\vec{y}_i^T = \frac{1}{n}\sum_{i=1}^{n}\left(U^T\vec{x}_i\right)\left(U^T\vec{x}_i\right)^T$$
$$= \frac{1}{n}\sum_{i=1}^{n}U^T\vec{x}_i\vec{x}_i^T U = U^T\left(\frac{1}{n}\sum_{i=1}^{n}\vec{x}_i\vec{x}_i^T\right)U$$
$$= U^T SU$$

Because $U^T SU$ is a diagonal matrix, PCA decorrelates the training samples $\vec{x}_i$ in the projected subspace. ■

Property 2.5: PCA maximizes the mutual information between $\vec{x}$ and $\vec{y} = U^T\vec{x}$ on Gaussian data.

Proof.

Let U be the projection, which is calculated according to PCA. Project all training samples $\vec{x}$ using U as $\vec{y} = U^T\vec{x}$, where $U \in R^{L\times L'}$.

The mutual information is $I(X,Y) = \int_Y\int_X p(\vec{x},\vec{y})\log\frac{p(\vec{x},\vec{y})}{p(\vec{x})\,p(\vec{y})}dXdY$ and we have $I(X,Y) = H(Y) - H(Y\,|\,X) = H(Y)$, because $\vec{y} = U^T\vec{x}$. Here, $H(Y)$ is the entropy of Y and $H(Y\,|\,X)$ is the entropy of Y with the given X.

The training samples are normally distributed with zero mean and covariance S, so the projected training samples are also normally distributed with zero mean and covariance $S_y = U^T S U$. Therefore, the entropy $H(\vec{y})$ is given by

$$H(\vec{y}) = -\int p(\vec{y}) \log(\vec{y}) d\vec{y}$$
$$= \frac{1}{2} + \frac{N'}{2}\log(2\pi) + \frac{1}{2}\log|S_y|$$
$$\leq \frac{1}{2} + \frac{N'}{2}\log(2\pi) + \frac{1}{2}\log\left|U^T S U\right|.$$

Because U consists of the eigenvectors of S corresponding to the first largest L' eigenvalues of S, U maximizes the mutual information on Gaussian data. ■

Observation 2.1: LDA maximizes the arithmetic mean of the KL divergences between all pairs of classes, under the assumption that the Gaussian distributions for different classes all have the same covariance matrix. The optimal projection matrix U in LDA can be obtained by maximizing a particular $V_\varphi(U)$ defined in (2.38).

Proof.

According to (2.36) and (2.37), the KL divergence between the *i*th class and the *j*th class in the projected subspace with the assumption of equal covariance matrices ($\Sigma_i = \Sigma_j = \Sigma$) is:

$$D_U\left(p_i \| p_j\right) = \operatorname{tr}\left(\left(U^T \Sigma U\right)^{-1} U^T D_{ij} U\right) + \text{constant}.$$

Then, we have

$$U^* = \arg\max_U \sum_{1 \leq i \neq j \leq c} q_i q_j D_U\left(p_i \| p_j\right)$$
$$= \arg\max_U \sum_{1 \leq i \neq j \leq c} \left(q_i q_j \operatorname{tr}\left(\left(U^T \Sigma U\right)^{-1} U^T D_{ij} U\right)\right)$$
$$= \arg\max_U \operatorname{tr}\left(\left(U^T \Sigma U\right)^{-1} U^T \left(\sum_{1 \leq i \neq j \leq c} q_i q_j D_{ij}\right) U\right)$$
$$= \arg\max_U \operatorname{tr}\left(\left(U^T \Sigma U\right)^{-1} U^T \left(\sum_{i=1}^{c-1} \sum_{j=i+1}^{c} q_i q_j D_{ij}\right) U\right),$$

where $q_i = n_i \Big/ \sum_{k=1}^{c} n_k$ is the prior probability of the *i*th class.

Because $S_b = \sum_{i=1}^{c-1} \sum_{j=i+1}^{c} q_i q_j D_{ij}$, as proved by Loog in [96], and $S_t = S_b + S_w = \Sigma$ where S_b, S_w, and S_t are defined in (2.10), we have:

$$\arg\max_U \sum_{1 \le i \ne j \le c} q_i q_j D_U\left(p_i \,\|\, p_j\right) = \arg\max_U \operatorname{tr}\left(\left(U^T S_w U\right)^{-1} U^T S_b U\right). \qquad \blacksquare$$

The generalized geometric mean is upper bounded by the arithmetic mean of the divergences, i.e.,

$$\prod_{1 \le i \ne j \le c} \left[D_U\left(p_i \,\|\, p_j\right)\right]^{\frac{q_i q_j}{\sum_{1 \le m \ne n \le c} q_m q_n}} \le \sum_{1 \le i \ne j \le c} \left(\frac{q_i q_j}{\sum_{1 \le m \ne n \le c} q_m q_n} D_U\left(p_i \,\|\, p_j\right)\right).$$

Proof.

Because $\sum_{1 \le i \ne j \le c} \frac{q_i q_j}{\sum_{1 \le m \ne n \le c} q_m q_n} = 1$ and $D_U\left(p_i \,\|\, p_j\right) > 0$, according to the *Jensen inequality* [21], we have

$$\sum_{1 \le i \ne j \le c} \frac{q_i q_j}{\sum_{1 \le m \ne n \le c} q_m q_n} D_U\left(p_i \,\|\, p_j\right)$$

$$= \sum_{1 \le i \ne j \le c} \frac{q_i q_j}{\sum_{1 \le m \ne n \le c} q_m q_n} \exp\left(\log\left(D_U\left(p_i \,\|\, p_j\right)\right)\right)$$

$$\ge \exp\left(\sum_{1 \le i \ne j \le c} \frac{q_i q_j}{\sum_{1 \le m \ne n \le c} q_m q_n} \log\left(D_U\left(p_i \,\|\, p_j\right)\right)\right)$$

$$= \prod_{1 \le i \ne j \le c} \left[D_U\left(p_i \,\|\, p_j\right)\right]^{\frac{q_i q_j}{\sum_{1 \le m \ne n \le c} q_m q_n}} .$$

$\blacksquare$

MGMD is a linear combination of 1) the log of the geometric mean of the divergences and 2) the log of the geometric mean of normalized divergences, i.e.,

$$U^* = \arg\max_U \left\{ \begin{array}{l} \alpha \log\left[\prod_{1 \le i \ne j \le c} E_U\left(p_i \,\|\, p_j\right)\right]^{\frac{1}{c(c-1)}} \\ +(1-\alpha) \log \prod_{1 \le i \ne j \le c} \left[D_U\left(p_i \,\|\, p_j\right)\right]^{\frac{q_i q_j}{\sum_{1 \le m \ne n \le c} q_m q_n}} \end{array} \right\}$$

where $0 < \alpha < 1$.

It is equivalent to

$$\arg\max_U \left\{ \sum_{1 \le i \ne j \le c} \log D_U\left(p_i \| p_j\right) - \eta \log\left(\sum_{1 \le i \ne j \le c} q_i q_j D_U\left(p_i \| p_j\right) \right) \right\},$$

where $\eta = \dfrac{\alpha c(c-1) \sum_{1 \le m \ne n \le c} q_m q_n}{c(c-1)(1-\alpha) + \alpha \sum_{1 \le m \ne n \le c} q_m q_n}$.

Deduction 2.1:

$$U^* = \arg\max_U \left\{ \begin{array}{l} \alpha \log \left[\prod_{1 \le i \ne j \le c} E_U\left(p_i \| p_j\right) \right]^{\frac{1}{c(c-1)}} \\ +(1-\alpha) \log \prod_{1 \le i \ne j \le c} \left[D_U\left(p_i \| p_j\right) \right]^{\frac{q_i q_j}{\sum_{1 \le m \ne n \le c} q_m q_n}} \end{array} \right\}$$

$$= \arg\max_U \left\{ \sum_{1 \le i \ne j \le c} \log D_U\left(p_i \| p_j\right) - \eta \log\left(\sum_{1 \le i \ne j \le c} q_i q_j D_U\left(p_i \| p_j\right) \right) \right\}.$$

Deduction

$$U^* = \arg\max_U \left\{ \begin{array}{l} \alpha \log \left[\prod_{1 \le i \ne j \le c} E_U\left(p_i \| p_j\right) \right]^{\frac{1}{c(c-1)}} \\ +(1-\alpha) \log \prod_{1 \le i \ne j \le c} \left[D_U\left(p_i \| p_j\right) \right]^{\frac{q_i q_j}{\sum_{1 \le m \ne n \le c} q_m q_n}} \end{array} \right\}$$

$$= \arg\max_U \left\{ \begin{array}{l} \alpha \dfrac{1}{c(c-1)} \sum_{1 \le i \ne j \le c} \log E_U\left(p_i \| p_j\right) \\ + \dfrac{(1-\alpha)}{\sum_{1 \le m \ne n \le c} q_m q_n} \sum_{1 \le i \ne j \le c} q_i q_j \log D_U\left(p_i \| p_j\right) \end{array} \right\}$$

$$= \arg\max_U \left\{ \begin{array}{l} \alpha \dfrac{1}{c(c-1)} \sum_{1 \le i \ne j \le c} \log \dfrac{q_i q_j D_U\left(p_i \| p_j\right)}{\sum_{1 \le m \ne n \le c} q_m q_n D_U\left(p_m \| p_n\right)} \\ + \dfrac{(1-\alpha)}{\sum_{1 \le m \ne n \le c} q_m q_n} \sum_{1 \le i \ne j \le c} q_i q_j \log D_U\left(p_i \| p_j\right) \end{array} \right\}$$

$$= \arg\max_U \left\{ \begin{array}{l} \left(\dfrac{\alpha}{c(c-1)} + \dfrac{(1-\alpha)}{\sum_{1\le m\ne n\le c} q_m q_n} \right) \sum_{1\le i\ne j\le c} \log q_i q_j D_U\left(p_i \,\|\, p_j\right) \\ -\alpha \log\left(\sum_{1\le m\ne n\le c} q_m q_n D_U\left(p_m \,\|\, p_n\right) \right) \end{array} \right\}$$

$$= \arg\max_U \left\{ \sum_{1\le i\ne j\le c} \log q_i q_j D_U\left(p_i \,\|\, p_j\right) - \eta \log\left(\sum_{1\le i\ne j\le c} q_i q_j D_U\left(p_i \,\|\, p_j\right) \right) \right\}$$

$$= \arg\max_U \left\{ \begin{array}{l} \sum_{1\le i\ne j\le c} \log q_i q_j + \sum_{1\le i\ne j\le c} \log D_U\left(p_i \,\|\, p_j\right) \\ -\eta \log\left(\sum_{1\le i\ne j\le c} q_i q_j D_U\left(p_i \,\|\, p_j\right) \right) \end{array} \right\}$$

$$= \arg\max_U \left\{ \sum_{1\le i\ne j\le c} \log D_U\left(p_i \,\|\, p_j\right) - \eta \log\left(\sum_{1\le i\ne j\le c} q_i q_j D_U\left(p_i \,\|\, p_j\right) \right) \right\},$$

where $\eta = \dfrac{\alpha c(c-1) \sum_{1\le m\ne n\le c} q_m q_n}{c(c-1)(1-\alpha) + \alpha \sum_{1\le m\ne n\le c} q_m q_n}$ and $0 < \alpha < 1$. The supremum of η is $c(c-1)$ and the infimum of η is 0. When $\alpha = 0 / \eta = 0$, (2.48) reduces to (2.43); and when $\alpha = 1 / \eta = c(c-1)$, (2.48) reduces to (2.46). ■

Claim 2.1: $L(U) = L(UB)$, where B is any orthogonal $r \times r$ matrix and U is the projection matrix, which maximizes $L(U)$ defined in (2.51).

Proof.

Because $L(U) = \sum_{1\le i\ne j\le c} \log KL_U\left(p_i \,\|\, p_j\right) - \eta \log\left(\sum_{1\le i\ne j\le c} q_i q_j KL_U\left(p_i \,\|\, p_j\right) \right)$, it is sufficient to prove $KL_U\left(p_i \,\|\, p_j\right) = KL_{UB}\left(p_i \,\|\, p_j\right)$. Here p_i is the probability density function for samples in the i^{th} class, i.e., $p_i = p_i(\vec{x}) = p(\vec{x} \mid y = i)$. According to the definition of $KL_U\left(p_i \,\|\, p_j\right)$ in (2.52), we have

$$\begin{aligned} KL_{UB}\left(p_i \,\|\, p_j\right) = \frac{1}{2} & \log\left|B^T U^T \Sigma_j UB\right| - \log\left|B^T U^T \Sigma_i UB\right| \\ & + \operatorname{tr}\left(\left(B^T U^T \Sigma_j UB\right)^{-1} \left(B^T U^T \Sigma_i UB\right) \right) \\ & + \operatorname{tr}\left(\left(B^T U^T \Sigma_j UB\right)^{-1} B^T U^T D_{ij} UB \right), \end{aligned}$$

where Σ_i is the covariance matrix of the i^{th} class and $D_{ij} = \left(\vec{m}_i - \vec{m}_j\right) \otimes \left(\vec{m}_i - \vec{m}_j\right)$.

Because $|AB| = |A||B|$ when A and B are square matrices, we have

$$
\begin{aligned}
KL_{UB}\left(p_i \,\|\, p_j\right) = \frac{1}{2}\log\left|U^T\Sigma_j U\right| - \log\left|U^T\Sigma_i U\right| \\
+ \operatorname{tr}\left(\left(B^T U^T \Sigma_j U B\right)^{-1}\left(B^T U^T \Sigma_i U B\right)\right) \\
+ \operatorname{tr}\left(\left(B^T U^T \Sigma_j U B\right)^{-1} B^T U^T D_{ij} U B\right).
\end{aligned}
$$

Because

$$
\begin{aligned}
\left(B^T U^T \Sigma_j U B\right)^{-1} &= B^{-1}\left(U^T\Sigma_j U\right)^{-1}\left(B^T\right)^{-1} \\
&= B^T\left(U^T\Sigma_j U\right)^{-1} B,
\end{aligned}
$$

we have

$$
\begin{aligned}
KL_{UB}\left(p_i \,\|\, p_j\right) = \frac{1}{2}\log\left|U^T\Sigma_j U\right| - \log\left|U^T\Sigma_i U\right| \\
+ \operatorname{tr}\left(\left(U^T \Sigma_j U\right)^{-1}\left(U^T \Sigma_i U\right)\right) \\
+ \operatorname{tr}\left(\left(U^T \Sigma_j U\right)^{-1} U^T D_{ij} U\right) \\
= KL_U\left(p_i \,\|\, p_j\right).
\end{aligned}
$$

Therefore, $L(U) = L(UB)$. ■

Lemma 2.1: If U is a solution to MGMKLD and $U = U_x \oplus U_x^{\perp}$, then U_x is a solution to MGMKLD, and $L(U) = L(U_x)$. Here, the column space of U_x is spanned by the samples $\left\{\vec{x}_{i;j}\right\}\Big|_{1\le i\le c}^{1\le j\le n_i}$ and the column space of $U_x^{\perp}$ is the orthogonal complement of the column space of U_x.

Proof.

Because $\left(U_x^{\perp}\right)^T \vec{x}_{i;j} = 0$ and $\left(U_x^{\perp}\right)^T U_x = 0$, we have

$$
\left(U_x + U_x^{\perp}\right)^T \vec{m}_i = U_x^T \vec{m}_i + \left(U_x^{\perp}\right)^T \vec{m}_i = U_x^T \vec{m}_i + \frac{1}{n_i}\sum_{j=1}^{n_i}\left(\left(U_x^{\perp}\right)^T \vec{x}_{i;j}\right) = U_x^T \vec{m}_i
$$

and

$$
\left(U_x \oplus U_x^{\perp}\right)^T \Sigma_i \left(U_x \oplus U_x^{\perp}\right)
$$

$$=\frac{1}{n_i}\sum_{j=1}^{n_i}\left(\left(\left(U_x\oplus U_x^{\perp}\right)^T\left(\vec{x}_{i;j}-\vec{m}_i\right)\right)\left(\left(U_x\oplus U_x^{\perp}\right)^T\left(\vec{x}_{i;j}-\vec{m}_i\right)\right)^T\right)$$

$$=\frac{1}{n_i}\sum_{j=1}^{n_i}\left(\left(U_x^T\left(\vec{x}_{i;j}-\vec{m}_i\right)\right)\left(U_x^T\left(\vec{x}_{i;j}-\vec{m}_i\right)\right)^T\right)=U_x^T\Sigma_iU_x.$$

That is $\left(U_x\oplus U_x^{\perp}\right)^T\vec{m}_i=U_x^T\vec{m}_i$ and $\left(U_x\oplus U_x^{\perp}\right)^T\Sigma_i\left(U_x\oplus U_x^{\perp}\right)=U_x^T\Sigma_iU_x$. With these two equations, we can get $KL_U\left(p_i\parallel p_j\right)=KL_{U_x}\left(p_i\parallel p_j\right)$, because:

$$KL_U\left(p_i\parallel p_j\right)$$

$$=\frac{1}{2}\begin{pmatrix}\log\left|U^T\Sigma_jU\right|-\log\left|U^T\Sigma_iU\right|\\+\operatorname{tr}\left(\left(U^T\Sigma_jU\right)^{-1}\left(U^T\left(\Sigma_i+D_{ij}\right)U\right)\right)\end{pmatrix}$$

$$=\frac{1}{2}\begin{pmatrix}\log\left|\left(U_x\oplus U_x^{\perp}\right)^T\Sigma_j\left(U_x\oplus U_x^{\perp}\right)\right|-\log\left|\left(U_x\oplus U_x^{\perp}\right)^T\Sigma_i\left(U_x\oplus U_x^{\perp}\right)\right|\\+\operatorname{tr}\begin{pmatrix}\left(\left(U_x\oplus U_x^{\perp}\right)^T\Sigma_j\left(U_x\oplus U_x^{\perp}\right)\right)^{-1}\\\left(\left(U_x\oplus U_x^{\perp}\right)^T\left(\Sigma_i+D_{ij}\right)\left(U_x\oplus U_x^{\perp}\right)\right)\end{pmatrix}\end{pmatrix}$$

$$=\frac{1}{2}\begin{pmatrix}\log\left|U_x^T\Sigma_jU_x\right|-\log\left|U_x^T\Sigma_iU_x\right|\\+\operatorname{tr}\left(\left(U_x^T\Sigma_jU_x\right)^{-1}\left(U_x^T\left(\Sigma_i+D_{ij}\right)U_x\right)\right)\end{pmatrix}=KL_{U_x}\left(p_i\parallel p_j\right)$$

Therefore $L(U)=L(U_x)$, i.e., the matrices U and $U_{\mathbf{x}}$ are equally good solutions to MGMKLD. ■

Deduction 2.2: To obtain the kernel Gram matrix [129] based representation in (2.61), we need to get the $U_\phi^T(U)\Sigma_{\phi;i}U_\phi$ reformulated by the kernel dot product trick as

$$U_\phi^T\Sigma_{\phi;i}U_\phi$$

$$=\frac{1}{n_i}\Lambda_{n\times H'}^TK_{C_i}\left(I_{C_i,C_i}-\frac{1}{n_i}1_{C_i,C_i}\right)\left(I_{C_i,C_i}-\frac{1}{n_i}1_{C_i,C_i}\right)^TK_{C_i}^T\Lambda_{n\times H'},$$

where $K_{:,j}$ is the $\left(\sum_{k=1}^{i-1}n_k+j\right)^{th}$ column of the kernel Gram matrix $K=\left[\phi\left(\vec{x}_{i;j}\right)\right]_{F\times n}^T\left[\phi\left(\vec{x}_{i;j}\right)\right]_{F\times n}=\left[k\left(\vec{x}_{i;j},\vec{x}_{i;j}\right)\right]$, $k\left(\vec{x}_{i;j},\vec{x}_{i;j}\right)$ is the kernel function [129], $I_{C_i,C_i}\in R^{C_i,C_i}$ is the identity matrix, $1_{C_i,C_i}\in R^{C_i,C_i}$ is a matrix in which every entry is 1, $K_{C_i} = \left[K_{.;j} \right]_{1 \le j \le n_i}$ is composed of the columns in the kernel Gram matrix from the $\left(\sum_{k=1}^{i-1} n_k + 1 \right)^{th}$ column to the $\left(\sum_{k=1}^{i} n_k \right)^{th}$ column.

Deduction

$$U_\phi^T \Sigma_{\phi;i} U_\phi$$

$$= \frac{1}{n_i} \sum_{j=1}^{n_i} \left(\begin{array}{c} \Lambda_{n \times H'}^T \left[\phi\left(\vec{x}_{p;q}\right) \right]_{F \times n}^T \left(\phi\left(\vec{x}_{i;j}\right) - \frac{1}{n_i} \sum_{k=1}^{n_i} \phi\left(\vec{x}_{i;k}\right) \right) \\ \left(\phi\left(\vec{x}_{i;j}\right) - \frac{1}{n_i} \sum_{l=1}^{n_i} \phi\left(\vec{x}_{i;l}\right) \right)^T \left[\phi\left(\vec{x}_{p;q}\right) \right]_{F \times n} \Lambda_{n \times H'} \end{array} \right)$$

$$= \Lambda_{n \times H'}^T \frac{1}{n_i} \left(\sum_{j=1}^{n_i} \left(K_{.;j} - \frac{1}{n_i} \sum_{k=1}^{n_i} K_{.;k} \right) \left(K_{.;j} - \frac{1}{n_i} \sum_{l=1}^{n_i} K_{.;l} \right)^T \right) \Lambda_{n \times H'}$$

$$= \Lambda_{n \times H'}^T \frac{1}{n_i} \left(\sum_{j=1}^{n_i} \left(K_{.;j} - \frac{1}{n_i} K_{C_i} 1_{C_i} \right) \left(K_{.;j} - \frac{1}{n_i} K_{C_i} 1_{C_i} \right)^T \right) \Lambda_{n \times H'}$$

$$= \Lambda_{n \times H'}^T \frac{1}{n_i} \left(K_{C_i} - \frac{1}{n_i} K_{C_i} 1_{C_i, C_i} \right) \left(K_{C_i} - \frac{1}{n_i} K_{C_i} 1_{C_i, C_i} \right)^T \Lambda_{n \times H'}$$

$$= \frac{1}{n_i} \Lambda_{n \times H'}^T K_{C_i} \left(I_{C_i, C_i} - \frac{1}{n_i} 1_{C_i, C_i} \right) \left(I_{C_i, C_i} - \frac{1}{n_i} 1_{C_i, C_i} \right)^T K_{C_i}^T \Lambda_{n \times H'} . \qquad \blacksquare$$

Deduction 2.3: To obtain the kernel Gram matrix [129] based representation in (2.61), we need to get the $U_\phi^T D_{\phi;ij} U_\phi$ reformulated by the kernel dot product trick as

$$U_\phi^T D_{\phi;ij} U_\phi$$

$$= \Lambda_{n \times H'}^T \left(\frac{1}{n_i} K_{C_i} 1_{C_i} - \frac{1}{n_j} K_{C_j} 1_{C_j} \right) \left(\frac{1}{n_i} K_{C_i} 1_{C_i} - \frac{1}{n_j} K_{C_j} 1_{C_j} \right)^T \Lambda_{n \times H'},$$

where $K_{.;j}$ is the $\left(\sum_{k=1}^{i-1} n_k + j \right)^{th}$ column of the kernel Gram matrix $K = \left[\phi\left(\vec{x}_{i;j}\right) \right]_{F \times n}^T \left[\phi\left(\vec{x}_{i;j}\right) \right]_{F \times n} = \left[k\left(\vec{x}_{i;j}, \vec{x}_{i;j}\right) \right]$, $k\left(\vec{x}_{i;j}, \vec{x}_{i;j}\right)$ is the kernel function [129], $I_{C_i, C_i} \in R^{C_i, C_i}$ is the identity matrix, $1_{C_i, C_i} \in R^{C_i, C_i}$ is the unit matrix,

$K_{C_i} = \left[K_{.;j} \right]_{1 \le j \le n_i}$ is composed of the columns in the kernel Gram matrix from the $\left(\sum_{k=1}^{i-1} n_k + 1 \right)^{th}$ column to the $\left(\sum_{k=1}^{i} n_k \right)^{th}$ column.

Deduction

$U_\phi^T D_{\phi;ij} U_\phi$

$$= \begin{pmatrix} \Lambda_{n\times H'}^T \left[\phi\left(\vec{x}_{p;q}\right) \right]_{H\times n}^T \left(\frac{1}{n_i} \sum_{k=1}^{n_i} \phi\left(\vec{x}_{i;k}\right) - \frac{1}{n_j} \sum_{l=1}^{n_j} \phi\left(\vec{x}_{j;l}\right) \right) \\ \left(\frac{1}{n_i} \sum_{k=1}^{n_i} \phi\left(\vec{x}_{i;k}\right) - \frac{1}{n_j} \sum_{l=1}^{n_j} \phi\left(\vec{x}_{j;l}\right) \right)^T \left[\phi\left(\vec{x}_{p;q}\right) \right]_{H\times n} \Lambda_{n\times H'} \end{pmatrix}$$

$$= \Lambda_{n\times H'}^T \left(\frac{1}{n_i} \sum_{k=1}^{n_i} K_{.;k} - \frac{1}{n_j} \sum_{l=1}^{n_j} K_{.;l} \right) \left(\frac{1}{n_i} \sum_{k=1}^{n_i} K_{.;k} - \frac{1}{n_j} \sum_{l=1}^{n_j} K_{.;l} \right)^T \Lambda_{n\times H'}$$

$$= \Lambda_{n\times H'}^T \left(\frac{1}{n_i} K_{C_i} 1_{C_i} - \frac{1}{n_j} K_{C_j} 1_{C_j} \right) \left(\frac{1}{n_i} K_{C_i} 1_{C_i} - \frac{1}{n_j} K_{C_j} 1_{C_j} \right)^T \Lambda_{n\times H'} . \qquad \blacksquare$$

Theorem 2.1: MGMKLD followed by KPCA is equal to KMGMKLD.

Proof.

To simplify the formulation, we assume that $\vec{m}_\phi = (1/n) \sum_{i=1}^{c} \sum_{j=1}^{n_i} \phi\left(\vec{x}_{i;j}\right) = 0$.

The eigenvectors calculated in KPCA are denoted as:

$$P = \left[\beta^{i;j} \right]_{\substack{1\le i\le c \\ 1\le j\le n_i}} \in R^{\sum_i n_i \times \sum_i n_i} .$$

Then,

$$K\beta^{i;j} = \lambda_{i;j} \beta^{i;j}, 1 \le i \le c; 1 \le j \le n_i .$$

For a sample $\mathbf{x}_{i;j}$, its corresponding vector in KPCA, the higher dimensional space is:

$$\vec{z}^{KPCA} = \left(\left[\phi\left(\vec{x}_{p;q}\right) \right]_{F\times n} P \right)^T \phi\left(\vec{x}_{i;j}\right) = P^T \left[k\left(\vec{x}_{p;q}, \vec{x}_{i;j}\right) \right]_{n\times 1} = P^T K_{.;j} .$$

Then, we have $U^T \Sigma_i^{KPCA} U$, given by:

$$U^T \Sigma_i^{KPCA} U = \frac{1}{n_i} U^T \sum_j \left(\vec{x}_{i;j}^{KPCA} - \vec{m}_i^{KPCA} \right) \left(\vec{x}_{i;j}^{KPCA} - \vec{m}_i^{KPCA} \right)^T U$$

$$= \frac{1}{n_i} U^T \sum_j \left(P^T K_{.;j} - \frac{1}{n_i} \sum_l P^T K_{.;j} \right) \left(P^T K_{.;j} - \frac{1}{n_i} \sum_l P^T K_{.;j} \right)^T U$$

$$=\frac{1}{n_i}U^TP^T\sum_j\left(K_{.;j}-\frac{1}{n_i}\sum_l K_{.;j}\right)\left(K_{.;j}-\frac{1}{n_i}\sum_l K_{.;j}\right)^T PU$$

$$=\frac{1}{n_i}U^TP^TK_{C_i}\left(I_{C_i,C_i}-\frac{1}{n_i}1_{C_i,C_i}\right)\left(I_{C_i,C_i}-\frac{1}{n_i}1_{C_i,C_i}\right)^T K_{C_i}^T PU$$

and $U^TD_{ij}^{KPCA}U$ is given by:

$$U^TD_{ij}^{KPCA}U=U^T\left(\frac{1}{n_i}\sum_{k=1}^{n_i}\vec{x}_{i;k}^{KPCA}-\frac{1}{n_j}\sum_{l=1}^{n_j}\vec{x}_{j;l}^{KPCA}\right)\left(\frac{1}{n_i}\sum_{k=1}^{n_i}\vec{x}_{i;k}^{KPCA}-\frac{1}{n_j}\sum_{l=1}^{n_j}\vec{x}_{j;l}^{KPCA}\right)^T U$$

$$=U^T\left(\frac{1}{n_i}\sum_{k=1}^{n_i}P^TK_{.;k}-\frac{1}{n_j}\sum_{l=1}^{n_j}P^TK_{.;l}\right)\left(\frac{1}{n_i}\sum_{k=1}^{n_i}P^TK_{.;k}-\frac{1}{n_j}\sum_{l=1}^{n_j}P^TK_{.;l}\right)^T U$$

$$=W^TP^T\left(\frac{1}{n_i}\sum_{k=1}^{n_i}K_{.;k}-\frac{1}{n_j}\sum_{l=1}^{n_j}K_{.;l}\right)\left(\frac{1}{n_i}\sum_{k=1}^{n_i}K_{.;k}-\frac{1}{n_j}\sum_{l=1}^{n_j}K_{.;l}\right)^T PU$$

$$=U^TP^T\left(\frac{1}{n_i}K_{C_i}1_{C_i}-\frac{1}{n_j}K_{C_j}1_{C_j}\right)\left(\frac{1}{n_i}K_{C_i}1_{C_i}-\frac{1}{n_j}K_{C_j}1_{C_j}\right)^T PU$$

As P is in full rank, denoting $\Lambda_{n\times H'}=PU$, we then have

$$U^T\Sigma_i^{KPCA}U=\Lambda_{n\times H'}^T\phi\left(\Sigma_i\right)\Lambda_{n\times H'}$$

and

$$U^TD_{ij}^{KPCA}U=\Lambda_{n\times H'}^T\phi\left(D_{ij}\right)\Lambda_{n\times H'}.$$

Therefore,

$$KL_{ij}\left(PU\right)=KL_{ij}\left(\Lambda_{n\times H'}\right)=KL_{ij}\left(\phi\left(U\right)\right).$$

Consequently,

$$L\left(PU\right)=L\left(\phi\left(U\right)\right).$$

That is MGMKLD followed by KPCA is equal to KMGMKLD. ■

Appendix for Chapter 3

Theorem 3.1 (Higher–Order Singular Value Decomposition)

A tensor $\mathbf{X} \in R^{L_1 \times L_2 \times \cdots \times L_M}$ can be decomposed as the product of a tensor $\mathbf{Y} \in R^{L_1 \times L_2 \times \cdots \times L_M}$ with a series of orthogonal matrices $U_k \in R^{L_k \times L_k}$, i.e.,

$$\mathbf{X} = \mathbf{Y} \prod_{k=1}^{M} \times_k U_k \, ,$$

such that, the subtensor of $\mathbf{Y}_{l_k = \alpha} \in R^{L_1 \times \cdots L_{k-1} \times L_{k+1} \times L_M}$, obtained by fixing the k^{th} ($1 \le l_k \le L_k$) index to α, is orthogonal to $\mathbf{Y}_{l_k = \beta} \in R^{L_1 \times \cdots L_{k-1} \times L_{k+1} \times L_M}$, i.e.,

$$\llbracket \mathbf{Y}_{l_k=\alpha} \otimes \mathbf{Y}_{l_k=\beta}; (1:M-1)(1:M-1) \rrbracket = 0 .$$

when $\alpha \ne \beta$.

Finally,

$$\left\| \mathbf{Y}_{l_k=1} \right\|_{Fro} \ge \left\| \mathbf{Y}_{l_k=2} \right\|_{Fro} \ge \cdots \ge \left\| \mathbf{Y}_{l_k=L_k} \right\|_{Fro} \ge 0 .$$

Proof.

Decompose the mode–*k* matricizing of $\mathbf{X}$ through SVD,

$$\text{mat}_k(\mathbf{X}) = U_k \Sigma_k V_k^T$$

where U_k, V_k are orthogonal matrices and $\Sigma_k = \text{diag}\{\sigma_k^1, \sigma_k^2, \cdots, \sigma_k^{L_k}\}$ with $\sigma_k^i \ge \sigma_k^j$ for all $i \le j$.

Then, we have

$$\text{mat}_k(\mathbf{X}) = U_k \, \text{mat}_k(\mathbf{Y}) \left[U_1 \otimes U_2 \otimes \cdots U_{k-1} \otimes U_{k+1} \otimes \cdots U_N \right]^T .$$

Therefore, we have

$$U_k \Sigma_k V_k^T = U_k \, \text{mat}_k(\mathbf{Y}) \left[U_1 \otimes U_2 \otimes \cdots U_{k-1} \otimes U_{k+1} \otimes \cdots U_N \right]^T .$$

i.e.,

$$\text{mat}_k(\mathbf{Y}) = \Sigma_k V_k^T \left[U_1 \otimes U_2 \otimes \cdots U_{k-1} \otimes U_{k+1} \otimes \cdots U_N \right].$$

Because U_k and V_k are unitary matrices,

$$\llbracket \mathbf{Y}_{l_k=\alpha} \otimes \mathbf{Y}_{l_k=\beta}; (1:M-1)(1:M-1) \rrbracket = 0 ,$$

for all $\alpha \ne \beta$, and

$$\left\|\mathbf{Y}_{l_k=1}\right\|_{Fro} = \sigma_k^1 \geq \left\|\mathbf{Y}_{l_k=2}\right\|_{Fro} = \sigma_k^2 \geq \cdots \geq \left\|\mathbf{Y}_{l_k=L_k}\right\|_{Fro} = \sigma_k^{L_k} \geq 0 .$$

It follows from the above analysis that HOSVD is proved. ■

Theorem 3.2

Minimizing $f\left(\vec{u}_k\,|_{k=1}^{M}\right) = \left\|\mathbf{X} - \lambda \prod_{k=1}^{M} {}_{\otimes} \vec{u}_k\right\|_{Fro}^2$ ($\lambda = \mathbf{X}\prod_{k=1}^{M} {}_{\times_k} \vec{u}_k^T$) is equivalent to maximizing

$$g\left(\vec{u}_k\,|_{k=1}^{M}\right) = \left\|\mathbf{X}\prod_{k=1}^{M} {}_{\times_k} \vec{u}_k^T\right\|_{Fro}^2 .$$

Moreover, $f = \left\|\mathbf{X}\right\|_{Fro}^2 - g$.

Proof.

Because

$$\begin{aligned} f\left(\vec{u}_k\,|_{k=1}^{M}\right) &= \left\|\mathbf{X} - \lambda \prod_{k=1}^{M} {}_{\otimes} \vec{u}_k\right\|_{Fro}^2 \\ &= \left\|\mathbf{X}\right\|_{Fro}^2 - 2\lambda \left\|\mathbf{X}\prod_{k=1}^{M} {}_{\times_k} \vec{u}_k^T\right\|_{Fro} + \lambda^2 \left\|\prod_{k=1}^{M} {}_{\otimes} \vec{u}_k\right\|_{Fro}^2 \\ &= \left\|\mathbf{X}\right\|_{Fro}^2 - \lambda^2 \\ &= \left\|\mathbf{X}\right\|_{Fro}^2 - g\left(\vec{u}_k\,|_{k=1}^{M}\right), \end{aligned}$$

minimizing $f\left(\vec{u}_k\,|_{k=1}^{M}\right)$ is equivalent to maximizing $g\left(\vec{u}_k\,|_{k=1}^{M}\right)$. ■

Theorem 3.3

Given a sequence of unitary matrices $U_k \in R^{L_k \times L'_k}$ ($1 \leq k \leq M$ and $L'_d < L_d$) and a tensor $\mathbf{X} \in R^{L_1 \times L_2 \times \cdots L_M}$, the function $f\left(\hat{\mathbf{X}}\right) = \left\|\mathbf{X} - \hat{\mathbf{X}}\right\|_{Fro}^2$, where $\operatorname{rank}_d\left(\hat{\mathbf{X}}\right) = L'_d$, is minimized, when $\hat{\mathbf{X}} \in R^{L_1 \times L_2 \times \cdots L_M}$ is given by

$$\hat{\mathbf{X}} = \left(\mathbf{X}\prod_{k=1}^{M} {}_{\times_k} U_k^T\right)\prod_{k=1}^{M} {}_{\times_k} U_k = \mathbf{X}\prod_{k=1}^{M} {}_{\times_k} \left(U_k U_k^T\right).$$

Proof.

It is sufficient to prove $\mathbf{Y}$ minimizes $g\left(\mathbf{Y}\right) = \left\|\mathbf{X} - \mathbf{Y}\prod_{k=1}^{M} {}_{\times_k} U_k\right\|_{Fro}^2$, where $\mathbf{Y} = \mathbf{X}\prod_{k=1}^{M} \times_k U_k^T$.

Let $\mathbf{E} = \mathbf{X} - \mathbf{Y}\prod_{k=1}^{M} \times_k U_k$.

Because the columns in U_k, $1 \le k \le M$, are orthogonal, we have

$$\begin{aligned}
\mathbf{E}\prod_{k=1}^{M} \times_k U_k^T &= \left(\mathbf{X} - \mathbf{Y}\prod_{k=1}^{M} \times_k U_k\right)\prod_{k=1}^{M} \times_k U_k^T \\
&= \mathbf{X}\prod_{k=1}^{M} \times_k U_k^T - \mathbf{Y}\prod_{k=1}^{M} \times_k U_k \prod_{k=1}^{M} \times_k U_k^T \\
&= \mathbf{X}\prod_{k=1}^{M} \times_k U_k^T - \mathbf{Y}\prod_{k=1}^{M} \times_k \left(U_k^T U_k\right) \\
&= \mathbf{X}\prod_{k=1}^{M} \times_k U_k^T - \mathbf{Y}.
\end{aligned}$$

Therefore, $\mathbf{Y}\prod_{k=1}^{M} \times_k U_k$ is the least square estimation of $\mathbf{X}$, i.e., $g(\mathbf{Y})$ is minimized when $\mathbf{Y} = \mathbf{X}\prod_{k=1}^{M} \times_k U_k^T$. ■

Theorem 3.4

For a given tensor $\mathbf{X} \in R^{L_1 \times L_2 \times \cdots L_M}$, minimizing

$$f\left(U_k |_{k=1}^{M}\right) = \left\| \mathbf{X} - \mathbf{X}\prod_{k=1}^{M} \times_k \left(U_k U_k^T\right) \right\|_{Fro}^2$$

is equivalent to maximizing

$$g\left(U_k |_{k=1}^{M}\right) = \left\| \mathbf{X}\prod_{k=1}^{M} \times_k U_k^T \right\|_{Fro}^2 .$$

Proof.

Because

$$\begin{aligned}
f\left(U_k |_{k=1}^{M}\right) &= \left\| \mathbf{X} - \left(\mathbf{X}\prod_{k=1}^{M} \times_k U_k^T\right)\prod_{k=1}^{M} \times_k U_k \right\|_{Fro}^2 \\
&= \|\mathbf{X}\|_{Fro}^2 - 2\left\langle \mathbf{X}, \left(\mathbf{X}\prod_{k=1}^{M} \times_k U_k^T\right)\prod_{k=1}^{M} \times_k U_k \right\rangle + \|\mathbf{Y}\|_{Fro}^2 \\
&= \|\mathbf{X}\|_{Fro}^2 - 2\left\langle \mathbf{X}\prod_{k=1}^{M} \times_k U_k^T, \mathbf{X}\prod_{k=1}^{M} \times_k U_k^T \right\rangle + \|\mathbf{Y}\|_{Fro}^2 \\
&= \|\mathbf{X}\|_{Fro}^2 - \|\mathbf{Y}\|_{Fro}^2 ,
\end{aligned}$$

where $\mathbf{Y} = \mathbf{X}\prod_{k=1}^{M} \times_k U_k^T$, minimizing $f\left(U_k \mid_{k=1}^{M}\right)$ is equivalent to maximizing $g\left(U_k \mid_{k=1}^{M}\right)$. ■

Deduction 3.1

$$U^* = \arg\max_U \left(\operatorname{tr}\left(U^T S_b U\right) - \zeta \operatorname{tr}\left(U^T S_w U\right) \right)$$

$$= \arg\max_U \left(\begin{array}{l} \operatorname{tr}\left(U^T \left[\sum_{i=1}^{c} n_i \left(m_i - m\right)\left(m_i - m\right)^T \right] U \right) \\ -\zeta \operatorname{tr}\left(U^T \left[\sum_{i=1}^{c} \sum_{j=1}^{n_i} \left(x_{i;j} - m_i\right)\left(x_{i;j} - m_i\right)^T \right] U \right) \end{array} \right)$$

$$= \arg\max_U \left(\begin{array}{l} \operatorname{tr}\left(\sum_{i=1}^{c} n_i U^T \left[\left(m_i - m\right)\left(m_i - m\right)^T \right] U \right) \\ -\zeta \operatorname{tr}\left(\sum_{i=1}^{c} \sum_{j=1}^{n_i} U^T \left[\left(x_{i;j} - m_i\right)\left(x_{i;j} - m_i\right)^T \right] U \right) \end{array} \right)$$

$$= \arg\max_U \left(\begin{array}{l} \sum_{i=1}^{c} n_i \operatorname{tr}\left(U^T \left[\left(m_i - m\right)\left(m_i - m\right)^T \right] U \right) \\ -\zeta \sum_{i=1}^{c} \sum_{j=1}^{n_i} \operatorname{tr}\left(U^T \left[\left(x_{i;j} - m_i\right)\left(x_{i;j} - m_i\right)^T \right] U \right) \end{array} \right)$$

$$= \arg\max_U \left(\begin{array}{l} \sum_{i=1}^{c} n_i \operatorname{tr}\left(U^T \left(m_i - m\right) \left[U^T \left(m_i - m\right) \right]^T \right) \\ -\zeta \sum_{i=1}^{c} \sum_{j=1}^{n_i} \operatorname{tr}\left(U^T \left(x_{i;j} - m_i\right) \left[U^T \left(x_{i;j} - m_i\right) \right]^T \right) \end{array} \right)$$

$$= \arg\max_U \left(\begin{array}{l} \sum_{i=1}^{c} n_i \llbracket \left(\left(m_i - m\right) \times_1 U^T \right) \otimes \left(\left(m_i - m\right) \times_1 U^T \right); (1)(1) \rrbracket \\ -\zeta \sum_{i=1}^{c} \sum_{j=1}^{n_i} \llbracket \left(\left(x_{i;j} - m_i\right) \times_1 U^T \right) \otimes \left(\left(x_{i;j} - m_i\right) \times_1 U^T \right); (1)(1) \rrbracket \end{array} \right)$$

Deduction 3.2

$$
U_l^* \big|_{l=1}^{M} = \arg\max_{U_l|_{l=1}^{M}} \left(\begin{array}{l}
\sum_{i=1}^{c} n_i \left[\begin{array}{l} \left(\left(\mathbf{M}_i - \mathbf{M} \right) \prod_{k=1}^{M} \times_k U_k^T \right) \\ \otimes \left(\left(\mathbf{M}_i - \mathbf{M} \right) \prod_{k=1}^{M} \times_k U_k^T \right) \end{array} ; (1:M)(1:M) \right] \\
- \zeta \sum_{i=1}^{c} \sum_{j=1}^{n_i} \left[\begin{array}{l} \left(\left(\mathbf{X}_{i;j} - \mathbf{M}_i \right) \prod_{k=1}^{M} \times_k U_k^T \right) \\ \otimes \left(\left(\mathbf{X}_{i;j} - \mathbf{M}_i \right) \prod_{k=1}^{M} \times_k U_k^T \right) \end{array} ; (1:M)(1:M) \right]
\end{array} \right)
$$

$$
= \arg\max_{U_l|_{l=1}^{M}} \left(\begin{array}{l}
\sum_{i=1}^{c} n_i \left[\begin{array}{l} \left(\left(\mathbf{M}_i - \mathbf{M} \right) \bar{\times}_l U_l^T \times_l U_l^T \right) \\ \otimes \left(\left(\mathbf{M}_i - \mathbf{M} \right) \bar{\times}_l U_l^T \times_l U_l^T \right) \end{array} ; (1:M)(1:M) \right] \\
- \zeta \sum_{i=1}^{c} \sum_{j=1}^{n_i} \left[\begin{array}{l} \left(\left(\mathbf{X}_{i;j} - \mathbf{M}_i \right) \bar{\times}_l U_l^T \times_l U_l^T \right) \\ \otimes \left(\left(\mathbf{X}_{i;j} - \mathbf{M}_i \right) \bar{\times}_l U_l^T \times_l U_l^T \right) \end{array} ; (1:M)(1:M) \right]
\end{array} \right)
$$

$$
= \arg\max_{U_l|_{l=1}^{M}} \left(\begin{array}{l}
\sum_{i=1}^{c} n_i \operatorname{tr} \left(U_l^T \left[\begin{array}{l} \left(\left(\mathbf{M}_i - \mathbf{M} \right) \bar{\times}_l U_l^T \right) \\ \otimes \left(\left(\mathbf{M}_i - \mathbf{M} \right) \bar{\times}_l U_l^T \right) \end{array} ; (\bar{l})(\bar{l}) \right] U_l \right) \\
- \zeta \sum_{i=1}^{c} \sum_{j=1}^{n_i} \operatorname{tr} \left(U_l^T \left[\begin{array}{l} \left(\left(\mathbf{X}_{i;j} - \mathbf{M}_i \right) \bar{\times}_l U_l^T \right) \\ \otimes \left(\left(\mathbf{X}_{i;j} - \mathbf{M}_i \right) \bar{\times}_l U_l^T \right) \end{array} ; (\bar{l})(\bar{l}) \right] U_l \right)
\end{array} \right)
$$

$$
= \arg\max_{U_l|_{l=1}^{M}} \operatorname{tr} \left(U_l^T \left(\begin{array}{l}
\sum_{i=1}^{c} \left[\begin{array}{l} n_i \operatorname{mat}_l \left(\left(\mathbf{M}_i - \mathbf{M} \right) \bar{\times}_l U_l^T \right) \\ \operatorname{mat}_l^T \left(\left(\mathbf{M}_i - \mathbf{M} \right) \bar{\times}_l U_l^T \right) \end{array} \right] \\
- \zeta \sum_{i=1}^{c} \sum_{j=1}^{n_i} \left[\begin{array}{l} \operatorname{mat}_l \left(\left(\mathbf{X}_{i;j} - \mathbf{M}_i \right) \bar{\times}_l U_l^T \right) \\ \operatorname{mat}_l^T \left(\left(\mathbf{X}_{i;j} - \mathbf{M}_i \right) \bar{\times}_l U_l^T \right) \end{array} \right]
\end{array} \right) U_l \right).
$$

Theorem 3.5 The alternating projection method based optimization procedure for GTDA converges.

Proof.

The alternating projection method never decreases the function value $f\left(U_l \big|_{l=1}^{M}\right)$ of GTDA between two successive iterations, because it can be interpreted as a type of monotonic algorithm. We define a continuous function

$$
f : S_1 \times S_2 \times \cdots \times S_M = \prod_{l=1}^{M} S_l \to R^+,
$$

where $U_l \in S_l$ and S_l is a closed set, which includes all possible U_l.

With the definition, f consists of M different mappings:

$$\begin{aligned} g\left(U_l\right) & \triangleq \underset{U_l \in S_l}{\arg\max}\, f\left(U_l \mid_{l=1}^{M}\right) \\ & = \underset{U_l \in S_l}{\arg\max}\, \operatorname{tr}\left(U_l^T \left(\begin{array}{l} \sum_{i=1}^{c} \left[\begin{array}{l} n_i \operatorname{mat}_l\left(\left(\mathbf{M}_i - \mathbf{M}\right) \bar{\times}_l U_l^T\right) \\ \operatorname{mat}_l^T\left(\left(\mathbf{M}_i - \mathbf{M}\right) \bar{\times}_l U_l^T\right) \end{array} \right] \\ -\zeta \sum_{i=1}^{c} \sum_{j=1}^{n_i} \left[\begin{array}{l} \operatorname{mat}_l\left(\left(\mathbf{X}_{i;j} - \mathbf{M}_i\right) \bar{\times}_l U_l^T\right) \\ \operatorname{mat}_l^T\left(\left(\mathbf{X}_{i;j} - \mathbf{M}_i\right) \bar{\times}_l U_l^T\right) \end{array} \right] \end{array} \right) U_l \right). \end{aligned}$$

The mapping can be calculated with the given $U_d \mid_{d=1}^{l-1}$ in the t^{th} iteration and $U_d \mid_{d=l+1}^{M}$ in the $(t-1)^{\text{th}}$ iteration of the for–loop in Steps 3–5 in Table 3.6.

Given an initial $U_1 \in S_1$, the alternating projection generates a sequence of items $\left\{\left(U_l\right)_t \mid_{l=1}^{M}\right\}$ via $U_l \in g\left(U_l\right)$, with each $l \in \left\{1, 2, \cdots M\right\}$. The sequence has the following relationship:

$$\begin{aligned} a = f(U_1,1) \le f(U_2,1) \le \cdots \le f(U_M,1) \le f(U_1,2) \le f(U_2,2) \le \\ \cdots \le f(U_1,t) \le f(U_2,t) \le \cdots \le f(U_1,T) \le f(U_2,T) \le \\ \cdots \le f(U_M,T) = b. \end{aligned}$$

where $T \to +\infty$, $f(U_l,t)$ means $f\left(\left(U_d\right)_t \mid_{d=1}^{l}, \left(U_d\right)_{t-1} \mid_{d=l+1}^{M}\right)$; and a and b are limited values in the R^+ space.

Formally, the alternating projection can be illustrated by a composition of M sub–algorithms defined as

$$\Omega_l : \left(U_l \mid_{l=1}^{M}\right) \mapsto \prod_{d=1}^{l-1} {}_{\times_d} U_d \times \operatorname{Map}\left(\vec{u}_l\right) \prod_{d=l+1}^{M} {}_{\times_d} U_d .$$

It follows that $\Omega \triangleq \Omega_1 \circ \Omega_2 \circ \cdots \circ \Omega_M$ is an algorithm for sets $S_l \mid_{l=1}^{M}$. All sub–algorithms $g\left(U_l\right)$ increase the value of f, so Ω is monotonic with respect to f. ■

References

[1]. M. Abramowitz and I. A. Stegun, (Eds.). *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, 9th printing. New York: Dover. 1972.

[2]. M. Aladjem, “Linear Discriminant Analysis for Two Classes via Removal of Classification Structure,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, no. 2, pp. 187–192, Feb. 1997.

[3]. P. Bartlett and J. Shawe–Taylor, “Generalization Performance of Support Vector Machines and Other Pattern Classifiers,” *Advances in Kernel Methods – Support Vector Learning*, B. Scholkopf, C. J. Burges, and A. J. Smola (eds.), MIT Press, Cambridge, USA, 1998.

[4]. P. Belhumeur and J. Hespanha and D. Kriegman, “Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 711–720, 1997.

[5]. M. Belkin and P. Niyogi, “Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering,” *Neural Information Processing Systems*, vol. 14, pp. 585–591, 2002.

[6]. M. Belkin and P. Niyogi, “Laplacian Eigenmaps for Dimensionality Reduction and Data Representation,” *Neural Computation*, vol. 15, no. 6, pp. 1,373–1,396, 2003.

[7]. R. E. Bellman, *Adaptive Control Processes*. Princeton University Press, Princeton, NJ, 1966.

[8]. Y. Bengio, J.–F. Paiement, P. Vincent, O. Delalleau, N. Le Roux, and M. Ouimet, “Out–of–Sample Extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering,” *Advances in Neural Information Processing Systems*, vol. 16, 2004.

[9]. A. Bissacco, A. Chiuso, Y. Ma and S. Soatto, “Recognition of Human Gaits,” *Proc. IEEE Int’l Conf. Computer Vision and Pattern Recognition*, vol. 2, pp. 52–57, 2001.

[10]. W. M. Boothby, *An Introduction to Differential Manifolds and Riemannian Geometry*, 2nd ed. San Diego, CA: Academic, 1986.

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

[12]. S. Boyd, S. J. Kim, L. Vandenberghe, and A. Hassibi, “A Tutorial on Geometric Programming,” *Optimization and Engineering*, 2006.

[13]. C. J. C. Burges, “Geometric Methods for Feature Extraction and Dimensional Reduction,” *In Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers*, Eds. L. Rokach and O. Maimon, Kluwer Academic Publishers, 2005.

[14]. L.M. Bregman, “The Relaxation Method to Find the Common Points of Convex Sets and Its Application to the Solution of Problems in Convex

Programming," *USSR Compt. Math. and Math. Phys.*, no. 7, pp. 200–217, 1967.

[15]. J.C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[16]. L. J. Buturovic, "Toward Bayes–Optimal Linear Dimension Reduction," *IEEE Trans. Pattern Analysis Machine Intelligence*, vol. 16, no. 4, pp. 420–424, April 1994.

[17]. N. Campbell, "Canonical Variate Analysis – a General Formulation," *Australian Journal of Statistics*, vol. 26, pp. 86–96, 1984.

[18]. H. Cevikalp, M. Neamtu, M. Wilkes, and A. Barkana, "Discriminative Common Vectors for Face Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*. vol. 27, no. 1, pp. 4–13, 2005.

[19]. L. F. Chen, H.Y. Liao, M. T. Ko, J. C. Lin, and G. J. Yu, "A New LDA–based Face Recognition System which Can Solve the Small Sample Size Problem," *Pattern Recognition*, vol. 33, no. 10, pp. 1,713–1,726, 2000.

[20]. Y. Chou, *Statistical Analysis*, Holt International, 1969.

[21]. T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.

[22]. D. Cunado, M. Nixon, and J. Carter, "Automatic Extraction and Description of Human Gait Models for Recognition Purposes," *Computer Vision and Image Understanding*, vol. 90, no. 1, pp. 1–41, 2003.

[23]. J. Cutting and L. Kozlowski, "Recognizing Friends by Their Walk: Gait Perception without Familiarity Cues," *Bulletin of the Psychonomic Society*, vol. 9, pp. 353 – 356, 1977.

[24]. L. S. Daniel and J. Weng, "Hierarchical Discriminant Analysis for Image Retrieval," *IEEE Trans. Pattern Analysis Machine Intelligence*, vol. 21, no. 5, pp. 386–401, May 1999.

[25]. J. G. Daugman, "Two–Dimensional Spectral Analysis of Cortical Receptive Field Profile," *Vision Research*, vol. 20, pp. 847–856, 1980.

[26]. J. G. Daugman, "Uncertainty Relation for Resolution in Space, Spatial Frequency, and Orientation Optimized by Two–Dimensional Visual Cortical Filters," *Journal of the Optical Society of America*, vol. 2, no. 7, pp. 1,160–1,169, 1985.

[27]. J. G. Daugman, "High Confidence Visual Recognition of Persons by a Test of Statistical Independence," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 15, no. 11, pp. 1,148–1,161, 1993.

[28]. F. De la Torre and T. Kanade "Multimodal Oriented Discriminant Analysis," *Int'l Conf. Machine Learning*, Bonn, Germany, Aug. 2005.

[29]. H. P. Decell and S. M. Mayekar, "Feature Combinations and the Divergence Criterion," *Computers and Math. With Applications*, vol. 3, pp. 71–76, 1977.

[30]. R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification*. John Wiley and Sons Inc. 2001.

[31]. S. Dudoit, J. Fridly, and T.P. Speed, “Comparison of Discrimination Methods for the Classification of Tumors Using Gene Expression Data”, *J. of the American Statistical Association*, vol. 97, no.457, pp. 77–87, 2002.

[32]. D. Dunn, W. E. Higgins, and J. Wakeley, “Texture Segmentation Using 2–D Gabor Elementary Functions,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 16, no. 2, pp. 130–149, 1994.

[33]. B. Efron, “Estimating the Error Rate of a Prediction Rule: Improvement on Cross–Validation,” *Journal of the American Statistical Association*, vol. 78, no. 382, pp. 316–331, 1983.

[34]. K. Etemad and R. Chellappa, “Discriminant Analysis for Recognition of Human Face Images,” *Journal of the Optical Society of America A*, vol. 14, no. 8, pp. 1,724–1,733, 1998.

[35]. L. P. Fatti and D. M. Hawkins, “Variable Selection in Heteroscedastic Discriminant Analysis,” *Journal of the American Statistical Association*, vol. 81, pp. 494-500, 1986.

[36]. M. Figueiredo and A.K. Jain, “Unsupervised Learning of Finite Mixture Models,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 3, pp. 381–396, Mar. 2002.

[37]. R. A. Fisher, “The Statistical Utilization of Multiple Samples,” *Ann. Eugenics*, vol. 8 pp. 376–386, 1938.

[38]. J. H. Friedman, “Regularized Discriminant Analysis,” *Journal of the American Statistical Association*, vol. 84, pp. 165–175, 1989.

[39]. K. Fukunaga, *Introduction to Statistical Pattern Recognition* (Second Edition), Academic Press, 1990.

[40]. K. Fukunaga and M. Mantock, “Nonparametric Discriminant Analysis,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 5, pp. 671–678, 1983.

[41]. G. Fung and O. L. Mangasarian, “Proximal Support Vector Machine Classifiers,” *Proc. 7th ACM SIGKDD Int’l Conf. Knowledge Discovery and Data Mining*, California, USA, pp. 77–86, 2001.

[42]. P. Gallinari, S. Thiria, F. Badran, and F. Folgelman–Soulie, “On the Relations between Discriminant Analysis and Multilayer Perceptrons,” *Neural Networks*, vol. 4, pp. 349–360, 1991.

[43]. A. Girgensohn and J. Foote, “Video Classification using Transform Coefficients,” Proc. *IEEE Int'l Conf. Acoustics, Speech, and Signal Processing*, vol. 6, pp. 3045–3048, 1999.

[44]. G. H. Golub and C. F. van Loan, *Matrix Computation* (Third Edition), The Johns Hopkins Univ. Press, 1996.

[45]. J. Han and B. Bhanu, “Statistical Feature Fusion for Gait–Based Human Recognition,” *Proc. IEEE Int’l Conf. Computer Vision and Pattern Recognition*, vol. 2, pp. 842–847, Washington, DC, 2004.

[46]. J. Han and B. Bhanu, “Individual Recognition Using Gait Energy Image,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, no. 2, pp. 316–322, 2006.

[47]. C.P. Hansen, *Rank–deficient and discrete ill–posed problems*. SIAM, Philadelphia, PA, 1997.

[48]. G. H. Hardy, J. E. Littlewood, and G. Pólya, *Inequalities*. Cambridge, England: Cambridge University Press, 1952.

[49]. T. Hastie, A. Buja, and R. Tibshirani, "Penalized Discriminant Analysis," *Annals of Statistics*, vol. 23, pp. 73–102, 1995.

[50]. T. Hastie and R. Tibshirani, "Flexible Discriminant by Mixture Models," *J. Royal Statistical Society* (*Series B*), vol. 58, pp. 155–176, 1996.

[51]. T. Hastie and R. Tibshirani, "Discriminant Analysis by Gaussian Mixtures," *Journal of the Royal Statistical Society Series B: Methodological*, vol. 58, pp. 155–176, 1996.

[52]. T. Hastie, R. Tibshirani, and A. Buja, "Flexible Discriminant Analysis by Optimal Scoring," *Journal of the American Statistical Association*, vol. 89, pp. 1,255–1,270, 1994.

[53]. T. Hastie, R. Tibshirani, and J.H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, New York: Springer, 2001.

[54]. B. Heisele, "Visual Object Recognition with Supervised Learning," *IEEE Intelligent Systems*, vol. 8, no. 3, pp. 38–42, 2003.

[55]. P. Howland and H. Park, "Generalizing Discriminant Analysis Using the Generalized Singular Value Decomposition," *IEEE Trans. Pattern Analysis Machine Intelligence*, vol. 26, no. 8, pp. 995–1,006, 2004.

[56]. L. Itti and C. Koch, "Computational Modeling of Visual Attention," *Nature Reviews Neuroscience*, vol. 2, no. 3, pp. 194–203, 2001.

[57]. L. Itti, C. Koch, and E. Niebur, "A Model of Saliency–based Visual Attention for Rapid Scene Analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1,254–1,259, 1998.

[58]. A. K. Jain, R. P. W. Duin, and J. Mao, "Statistical Pattern Recognition: A Review," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 4–37, 2000.

[59]. A. K. Jain, S. Prabhakar, and L. Hong, "A Multichannel Approach to Fingerprint Classification," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 21, no. 4, pp. 348–359, 1999.

[60]. E.T. Jaynes, *Probability Theory: The Logic of Science* (Principles and Elementary Applications), Cambridge University Press, 2003.

[61]. B. Jelinek, "Review on Heteroscedastic Discriminant Analysis," Unpublished Report. http://www.cavs.msstate.edu/hse/ies/publications/courses/ece_8443/papers/2001/hda2/p01_paper_v00.pdf#search=%22%22Review%20on%20Heteroscedastic%20Discriminant%20Analysis%22%22

[62]. X. Jing and D. Zhang, "A Face and Palmprint Recognition Approach based on Discriminant DCT Feature Extraction," *IEEE Trans. Systems, Man and Cybernetics*, Part B, vol. 34, no. 6, pp. 2,405–2,415, 2004.

[63]. A.Y. Johnson and A.F. Bobick, "A Multi–View Method for Gait

Recognition using Static Body Parameters," *Proc. IEEE Int'l Conf. on Audio– and Video– Based Biometric Person Authentication*, pp. 301–311, 2001.

[64]. I. T. Jolliffe, *Principal Component Analysis* (2nd edition), Springer, 2002.

[65]. A. Kale, A. N. Rajagopalan, N. Cuntoor, and V. Kruger, "Gait–based Recognition of Humans using Continuous HMMs," *Proc. IEEE Int'l Conf. Automatic Face and Gesture Recognition*, pp. 321–326, Washington, DC, 2002.

[66]. A. Kale, A. Sundaresan, A. N. Rajagopalan, N. P. Cuntoor, A. K. Roy–Chowdhury, V. Kruger, and R. Chellappa, "Identification of Humans using Gait," *IEEE Trans. Image Processing*, vol. 13, no. 9, pp. 1,163–1,173, 2004.

[67]. M. Katz, H.–G. Meier, H. Dolfing, and D. Klakow, "Robustness of Linear Discriminant Analysis in Automatic Speech Recognition," *Proc. IEEE. Int'l Conf. on Pattern Recognition*, vol. 3, 2002.

[68]. T.K. Kim and J. Kittler, "Locally Linear Discriminant Analysis for Multimodally Distributed Classes for Face Recognition with a Single Model Image," *IEEE Trans. Pattern Analysis Machine Intelligence*, vol. 27, no. 3, pp. 318–327, Mar., 2005.

[69]. S.J. Kim, A. Magnani, and S. Boyd, "Robust Fisher Discriminant Analysis," *Advances in Neural Information Processing Systems*, Vancouver, British Columbia, 2005.

[70]. N. Kumar and A. G. Andreou, "Heteroscedastic Discriminant Analysis and Reduced Rank HMMs for Improved Speech Recognition," *Speech Communcation*, vol. 26, pp. 283–297, 1998.

[71]. S. Kung, M. Mak, and S. Lin, *Biometric Authentication*. Prentice Hall, 2004.

[72]. P. A. Lachenbruch, *Discriminant Analysis*. Hafner Press, New York, 1975.

[73]. G. Lanckriet, N. Cristianini, P. Bartlett, L. Ghaoui, and M. Jordan, "Learning the Kernel Matrix with Semidefinite Programming," *J. of Machine Learning Research*, vol. 5, pp. 27–72, 2004.

[74]. G. Lanckriet, L. Ghaoui, C. Bhattacharyya, and M. Jordan, "A Robust Minimax Approach to Classification", *J. of Machine Learning Research*, vol. 3, pp. 555–582, 2002.

[75]. L. D. Lathauwer, Signal Processing Based on Multilinear Algebra, Ph.D. Thesis, Katholike Universiteit Leuven, 1997.

[76]. L. De Lathauwer, B. De Moor, and J. Vandewalle, "A Multilinear Singular Value Decomposition," *SIAM J. on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253–1278, 2001.

[77]. L. De Lathauwer, B. De Moor, and J. Vandewalle, "On the Best Rank–1 and Rank–(r1,r2,...,rn) Approximation of Higher–Order Tensors," *SIAM J. on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1,324–1,342, 2000.

[78]. L. Lee, G. Dalley, and K. Tieu, "Learning Pedestrian Models for Silhouette Refinement," *Proc. IEEE Int'l Conf. Computer Vision*, vol. 1,

pp. 663–670, Nice, France, 2003.

[79]. L. Lee and W. E. L. Grimson, "Gait Analysis for Recognition and Classification," *Proc. IEEE Int'l Conf. Automatic Face and Gesture Recognition*, pp. 155–162, Washington, DC, 2002.

[80]. T. S. Lee, "Image Representation Using 2D Gabor Wavelets," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, no. 10, pp. 959–971, 2003.

[81]. Y. Leedan and P. Meer, "Heteroscedastic Regression in Computer Vision: Problems with Bilinear Constraint," *International Journal of Computer Vision*, vol. 37, no. 2, pp. 127-150, 2000.

[82]. A. Leonardis and H. Bischof, Robust Recovery of Eigenimages in the Presence of Outliers and Occlusions, *Journal of Computing and Information Technology*, vol. 4, no. 1, pp. 25–36, 1996.

[83]. Q. Li, *An Integration Framework of Feature Selection and Extraction for Appearance based Recognition*, PhD Thesis, UDEL1472, University of Delaware, 2006.

[84]. S. Z. Li, X. Lu, X. Hou, X. Peng, and Q. Cheng, "Learning Multiview Face Subspaces and Facial Pose Estimation using Independent Component Analysis," *IEEE Trans. Image Processing*, vol. 14, no. 6, pp. 705–712, 2005.

[85]. T. Li, M. Ogihara, Q. Li, "A Comparative Study on Content–based Music Genre Classification," *Proc. Annual ACM Conf. on Research and Development in Information Retrieval*, pp. 282–289, 2003.

[86]. J. J. Little and J. E. Boyd, "Recognizing People by Their Gait: the Shape of Motion," *Videre*, vol. 1, no. 2, pp. 1–32, 1998.

[87]. C. Liu, "Gabor–based Kernel PCA with Fractional Power Polynomial Models for Face Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 572–581, 2004.

[88]. C. Liu and H. Wechsler, "Gabor Feature Based Classification Using the Enhanced Fisher Linear Discriminant Model for Face Recognition," *IEEE Trans. Image Processing*, vol. 11, no. 4, pp. 467–476, 2002.

[89]. C. Liu and H. Wechsler, "Enhanced Fisher Linear Discriminant Models for Face Recognition," *Proc. IEEE Int'l Conf. Pattern Recognition*, vol.2, pp. 1,368–1,372, Brisbane, Australia, 1998.

[90]. X. Liu, A. Srivastava, and K. Gallivan, "Optimal Linear Representations of Images for Object Recognition," *IEEE Trans. Pattern Analysis Machine Intelligence*, vol. 26, no. 5, pp. 662–666, May 2004.

[91]. Z. Liu and S. Sarkar, "Simplest Representation yet for Gait Recognition: Averaged Silhouette," *Proc. IEEE Int'l Conf. Pattern Recognition*, vol. 4, pp. 211–214, Cambridge, England, 2004.

[92]. Z. Liu and S. Sarkar, "Improved Gait Recognition by Gait Dynamics Normalization," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, no. 6, pp. 863 – 876, 2006.

[93]. M. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret, "Applications of Second–Order Cone Programming," *Linear Algebra and its Applications*,

vol. 284, pp. 193–228, 1998.

[94]. R. Lotlikar and R. Kothari, "Adaptive Linear Dimensionality Reduction for Classification," *Pattern Recognition*, vol. 33, no. 2, pp. 185–194, February 2000.

[95]. R. Lotlikar and R. Kothari, "Fractional–Step Dimensionality Reduction," *IEEE Trans. Pattern Analysis Machine Intelligence*, vol. 22, no. 6, pp. 623–627, June 2000.

[96]. M. Loog, "Approximate Pairwise Accuracy Criteria for Multiclass Linear Dimension Reduction: Generalizations of the Fisher Criterion," Delft Univ. Technique Report, 1999.

[97]. M. Loog, Supervised Dimensionality Reduction and Contextual Pattern Recognition in Medical Image Processing, Ph.D. thesis, Image Sciences Institute, Utrecht University, 2004.

[98]. M. Loog, R. P.W. Duin, and R. Haeb–Umbach, "Multiclass Linear Dimension Reduction by Weighted Pairwise Fisher Criteria," *IEEE Trans. Pattern Analysis Machine Intelligence*, vol. 23, no. 7, pp. 762–766, July 2001.

[99]. M. Loog and R. P.W. Duin, "Linear Dimensionality Reduction via a Heteroscedastic Extension of LDA: The Chernoff Criterion," *IEEE Trans. Pattern Analysis Machine Intelligence*, vol. 26, no. 6, pp. 732–739, June 2004.

[100]. J. Lu, K.N. Plataniotis, and A.N. Venetsanopoulos, "Face Recognition Using LDA Based Algorithms," *IEEE Trans. Neural Networks*, vol. 14, no. 1, pp. 195–200, Jan. 2003.

[101]. S. Marcelja, "Mathematical Description of the Responses of Simple Cortical Cells," *Journal of the Optical Society of America*, vol. 70, no. 11, pp. 1297–1300, 1980.

[102]. A. Marshall and I. Olkin, "Multivariate Chebyshev Inequalities," *Annals of Mathematical Statistics*, vol. 31, no. 4, pp. 1,001–1,014, 1960.

[103]. G.J. McLachlan, *Discriminant Analysis and Statistical Pattern Recognition*, Wiley–Interscience, New York, 1992.

[104]. S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. R. Muller, "Fisher Discriminant Analysis with Kernels," *IEEE Workshop on Neural Networks for Signal Processing*, pp. 41–48, 1999.

[105]. T. Mitchell, *Machine Learning*, McGraw Hill, New York, 1997.

[106]. D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis* (4th Edition), John Wiley & Sons, 2006.

[107]. H. Moon and P.J. Phillips, "Computational and Performance Aspects of PCA–based Face Recognition Algorithms," *Perception*, vol. 30, pp. 303–321, 2001.

[108]. T. K. Moon, P. Howland, J. H. Gunther, "Document Author Classification using Generalized Discriminant Analysis," *Proc. Workshop on Text Mining, SIAM Int'l Conf. on Data Mining*, 2006.

[109]. K.R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf, "An

Introduction to Kernel–Based Learning Algorithms," *IEEE Trans. Neural Networks*, vol. 12, no. 2, pp. 181–201, Feb. 2001.

[110]. N. Murata, T. Takenouchi, T. Kanamori, and S. Eguchi, "Information Geometry of U–Boost and Bregman Divergence," *Neural Computation*, vol. 16, no. 7, pp. 1,437–1,481, July, 2004.

[111]. M. Murray, "Gait as a Total Pattern of Movement," *American Journal of Physical Medicine*, vol. 16, pp. 290 – 332, 1967.

[112]. M. Murray, A. Drought, and R. Kory, "Walking Pattern of Normal Men," *J. of Bone and Joint Surgery*, vol. 46, no. A(2), pp. 335–360, 1964.

[113]. S. Nikolopoulos, S. Zafeiriou, P. Sidiropoulos, N. Nikolaidis, and I. Pitas, "Image Replica Detection Using R–Trees and Linear Discriminant Analysis," *Proc. IEEE Int'l Conf. Multimedia and Expo*, 2006.

[114]. J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, 1999.

[115]. D. G. Northcutt, *Multilinear Algebra*, Cambridge U Press, 1984.

[116]. X.M. Pardo, P. Radeva, and D. Cabello, "Discriminant Snakes for 3D Reconstruction of Anatomical Organs," *Medical Image Analysis*, vol. 7, no. 3, pp. 293–310, 2003.

[117]. J.P. Pedroso and N. Murata, "Support Vector Machines for Linear Programming: Motivation and Formulations," BSIS Technical Report No.99–XXX, RIKEN Brain Science Institute, Japan, 1999.

[118]. P. J. Phillips, H. Moon, S. A. Rizvi, and P. J. Rauss, "The FERET Evaluation Methodology for Face–Recognition Algorithms," *IEEE Trans. on Pattern Recognition and Machine Intelligence*, vol. 22, no. 10, pp. 1090–1104, 2000.

[119]. R. Plamondon and S. N. Srihari On–Line and Off–Line Handwriting Recognition: A Comprehensive Survey. *IEEE Trans. on Pattern Recognition and Machine Intelligence*, vol. 22, no. 1, pp. 63–84, 2000.

[120]. I. Popescu and D. Bertsimas, "Optimal Inequalities in Probability Theory: a Convex Optimization Approach," Technique Report TM62, Insead.

[121]. B.G. Prasad, K.K. Biswas, and S.K. Gupta, "Region–based Image Retrieval using Integrated Color, Shape, and Location Index," *Computer Vision and Image Understanding*, vol. 94, no. 1–3, pp. 192–233, 2004.

[122]. C. R. Rao, "The Utilization of Multiple Samples in Problems of Biological Classification," *J. Royal Statistical Soc., B*, vol. 10, pp. 159–203, 1948.

[123]. S. Raudys and R. P. W. Duin, "On Expected Classification Error of the Fisher Linear Classifier with Pseudo–inverse Covariance Matrix," *Pattern Recognition Letter*, vol. 19, no. 5–6, 1998.

[124]. S. T. Roweis and L. K. Saul, "Nonlinear Dimensionality Reduction by Locally Linear Embedding," *Science*, vol. 290, pp. 2,323–2,326, 2000.

[125]. Y. Rui, T. S. Huang, and S. F. Chang, "Image Retrieval: Current Techniques, Promising Directions and Open Issues," *Journal of Visual Communication and Image Representation*, vol. 10, pp. 39–62, 1999.

[126]. S. Sarkar, P. Phillips, Z. Liu, I. Vega, P. Grother, and K. Bowyer, "The HumanID Gait Challenge Problem: Data Sets, Performance, and

Analysis," *IEEE Trans. Pattern Analysis Machine Intelligence*, vol. 27, no. 2, pp. 162–177, 2005.

[127]. L. K. Saul and S. T. Roweis, "Think Globally, Fit Locally: Unsupervised Learning of Low Dimensional Manifolds," *Journal of Machine Learning Research*, vol. 4, pp. 119–155, 2003.

[128]. B. Schölkopf and A. Smola, *Learning with Kernels: Support Vector Machines, Regularization*, Optimization, and Beyond (Adaptive Computation and Machine Learning), MIT Press, 2001.

[129]. B. Schölkopf, A. J. Smola, and K.R. Müller, "Kernel Principal Component Analysis," *Advances in Kernel Methods: Support Vector Learning*, MIT Press, pp. 327–352, Cambridge, MA, USA, 1999.

[130]. B. Schölkopf, A. Smola, R. C. Williamson, and P. L. Bartlett, "New Support Vector Algorithms," *Neural Computation*, vol. 12, pp. 1,207 – 1,245, 2000.

[131]. J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, Aug. 2000.

[132]. A. Shashua and A. Levin, "Linear Image Coding for Regression and Classification Using the Tensor–rank Principle," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, vol. 1, pp. 42–49, 2001.

[133]. J. R. Smith and S.–F. Chang, "Transform Features for Texture Classification and Discrimination in Large Image Databases," *Proc. IEEE Int'l Conf. Image Processing*, vol. 3, pp. 407–411, 1994.

[134]. A. Smola, T.T. Friess, and B. Schölkopf, "Semiparametric Support Vector and Linear Programming Machines," *Neural and Information Processing Systems*, vol. 11, 1999.

[135]. T.R. Strohmann, A. Belitski, G.Z. Grudic, and D. DeCoste, "Sparse Greedy Minimax Probability Machine Classification," *Advances in Neural Information Processing Systems*, Vancouver and Whistler, British Columbia, 2003.

[136]. J. Sun, D. Tao, and C. Faloutsosy, "Beyond Streams and Graphs: Dynamic Tensor Analysis," *Proc. ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, 2006.

[137]. J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle, *Least Squares Support Vector Machines*, World Scientific, 2002.

[138]. J.A.K. Suykens, J. Vandewalle, "Least Squares Support Vector Machine Classifiers," *Neural Processing Letters*, vol. 9, no. 3, pp. 293–300, 1999.

[139]. D. L. Swets and J. Weng, "Using Discriminant Eigenfeatures for Image Retrieval," *IEEE Trans. Pattern Analysis and Machine Intelligences*, vol. 18, no. 8, pp. 831–836, 1996.

[140]. Y. Sun and R. Fisher, "Object–based Visual Attention for Computer Vision," *Artificial Intelligence*, vol. 146, no. 1, pp. 77–123, 2003.

[141]. J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle, *Least Squares Support Vector Machines*, World Scientific,

Singapore.

[142]. R. Tanawongsuwan and A. Bobick, “Modelling the Effects of Walking Speed on Appearance–based Gait Recognition,” *Proc. IEEE Int’l Conf. Computer Vision and Pattern Recognition*, 2, 783–790, 2004.

[143]. D. Tao, X. Li, X. Wu, and S. J. Maybank, “Elapsed Time in Human Gait Recognition: A New Approach,” *Proc. IEEE Int’l Conf. on Acoustics, Speech, and Signal Processing*, vol. 2, pp. 177–180, 2006.

[144]. D. Tao, X. Li, X. Wu, and S. J. Maybank, “Human Carrying Status in Visual Surveillance,” *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, pp. 1,670–1,677, 2006.

[145]. D. Tao, X. Li, X. Wu, and S. J. Maybank, “Tensor Rank One Discriminant Analysis,” Submitted to *IEEE Trans. on Pattern Analysis and Machine Intelligence*. (Under Major Revision)

[146]. D. Tao, X. Li, X. Wu, and S. J. Maybank, “General Averaged Divergences Analysis,” Submitted to *IEEE Trans. on Pattern Analysis and Machine Intelligence*. (Under Major Revision)

[147]. D. Tao, X. Li, X. Wu, and S. J. Maybank, “General Tensor Discriminant Analysis and Gabor Features for Gait Recognition,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2007. (To appear)

[148]. D. Tao, X. Li, and S. J. Maybank, “Negative Samples Analysis in Relevance Feedback,” *IEEE Trans. on Knowledge and Data Engineering*, 2007. (To appear)

[149]. D. Tao, X. Li, W. Hu, S. J. Maybank, and X. Wu, “Supervised Tensor Learning: A Framework,” *Knowledge and Information Systems* (Springer), 2007. (To appear)

[150]. D. Tao, X. Li, W. Hu, S. J. Maybank, and X. Wu, “Supervised Tensor Learning,” *Proc. IEEE Int’l Conf. on Data Mining*, pp. 450–457, 2005.

[151]. D. Tao, X. Li, W. Hu, and S. J. Maybank, “Stable Third–Order Tensor Representation for Color Image Classification,” *Proc. IEEE Int’l Conf. on Web Intelligence*, pp. 641–644, 2005.

[152]. D. Tao, X. Tang, X. Li, and Y. Rui, “Kernel Direct Biased Discriminant Analysis: A New Content–based Image Retrieval Relevance Feedback Algorithm,” *IEEE Trans. on Multimedia*, vol. 8, no. 4, pp. 716–727, 2006.

[153]. D. Tao and X. Tang, “Kernel Full–space Biased Discriminant Analysis,” *Proc. IEEE Int’l Conf. on Multimedia and Expo*, pp. 1,287–1,290, 2004.

[154]. D. Tao and X. Tang, “A Direct Method to Solve the Biased Discriminant Analysis in Kernel Feature Space for Content–based Image Retrieval,” *Proc. IEEE Int’l Conf. on Acoustics, Speech, and Signal Processing*, pp. 441–444, 2004.

[155]. J. B. Tenenbaum, V. de Silva, and J. C. Langford, “A Global Geometric Framework for Nonlinear Dimensionality Reduction,” *Science*, vol. 290, pp. 2,319–2,323, 2000.

[156]. A. B. Torralba and A. Oliva, “Semantic Organization of Scenes using Discriminant Structural Templates,” *Proc. IEEE Int'l Conf. Computer Vision*, pp. 1,253–1,258, 1999.

[157]. A.M. Treisman and G. Gelade, "A Feature–Integration Theory of Attention," *Cognitive Psychology*, vol. 12, no. 1, pp. 97–136, 1980.

[158]. M. Turk and A. Pentland, "Eigenfaces for Recognition," *J. of Cognitive Neurosicence*, vol. 3, no. 1, pp. 71–86, 1991.

[159]. L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM Review*, vol. 38, no. 1, pp. 49–95, 1996.

[160]. R. Vanderbei, *Linear Programming: Foundations and Extensions* (2nd edition), Springer, 2001.

[161]. V. Vapnik, *The Nature of Statistical Learning Theory*, Springer–Verlag, New York, 1995.

[162]. M. A. O. Vasilescu and D. Terzopoulos, "Multilinear Subspace Analysis for Image Ensembles," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, vol.2, pp. 93–99, Madison, WI, 2003.

[163]. G. Veres, L. Gordon, J. Carter, and M. Nixon, "What Image Information is Important in Silhouette–based Gait Recognition?," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, vol. 2, pp. 776–782, 2004.

[164]. J. Z. Wang, L. Li, and G. Wiederhold, "SIMPLIcity: Semantics–Sensitive Integrated Matching for Picture Libraries," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 23, no. 9, pp. 947–963, 2001.

[165]. L. Wang, H. Ning, T. Tan, and W. Hu, "Fusion of Static and Dynamic Body Biometrics for Gait Recognition," *Proc. IEEE Int'l Conf. Computer Vision*, 2, 2003.

[166]. L. Wang, T. Tan, H. Ning, and W. Hu, "Silhouette Analysis–based Gait Recognition for Human Identification," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 25, no. 12, pp. 1,505–1,518, 2003.

[167]. L. Wang, Y. Zhang, and J. Feng, "On the Euclidean Distance of Images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1,334–1,339, 2005.

[168]. H. Wechslet, J. Phillips, V. Bruse, F. Soulie, and T. Hauhg, editors. *Face Recognition: From Theory to Application*. Springer–Verlag, Berlin, 1998.

[169]. K. Q. Weinberger, J. Blitzer, and L. K. Saul, "Distance Metric Learning for Large Margin Nearest Neighbor Classification," *Neural Information Processing Systems*, 2005.

[170]. W. L. Winston, J. B. Goldberg, and M. Venkataramanan, *Introduction to Mathematical Programming: Operations Research* (4th edition), Duxbury Press, 2002.

[171]. D. Xu, S. Yan, L. Zhang, H.–J. Zhang, Z. Liu, and H.–Y. Shum, "Concurrent Subspaces Analysis," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, vol. 2, pp. 203–208, 2005.

[172]. S. Yan, D. Xu, B. Zhang, and H.J. Zhang, "Graph Embedding: A General Framework for Dimensionality Reduction," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, vol. 2 pp. 830–837, 2005.

[173]. J. Ye, R. Janardan, and Q. Li, "Two–Dimensional Linear Discriminant Analysis," *Neural Information Processing Systems*, pp. 1,569–1,576,

Vancouver, Canada, 2005.

[174]. J. Ye and Q. Li, "A Two–Stage Linear Discriminant Analysis via QR–Decomposition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, no. 6, pp. 929–941, 2005.

[175]. H. Yu and J. Yang, "A Direct LDA Algorithm for High–dimensional Data with Application to Face Recognition," *Pattern Recognition*, vol. 34, no. 12, pp. 2,067–2,070, Dec. 2001.

[176]. S. Yu, D. Tan, and T. Tan, "Modelling the Effect of View Angle Variation on Appearance–Based Gait Recognition," *Proc. Asian Conf. Computer Vision*, vol. 1, pp. 807–816, 2006.

[177]. S. X. Yu and J. Shi, "Multiclass Spectral Clustering," *Proc. IEEE Int'l Conf. Computer Vision*, pp. 313–319, 2003.

[178]. W.I. Zangwill, *Nonlinear Programming: A Unified Approach*. Englewood Cliffs, NJ: Prentice–Hall, 1969.

[179]. D. Zhang, A. W.–K. Kong, J. You, and M. Wong, "Online Palmprint Identification," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 25, no. 9, pp. 1,041–1,050, 2003.

[180]. P. Zhang, J. Peng and N. Riedel, "Discriminant Analysis: A Least Squares Approximation View," *IEEE Workshop on Learning in conjunction with IEEE Int'l Conf. Computer Vision and Pattern Recognition*, San Diego, CA, 2005.

[181]. X. Zhang, *Matrix Analysis and Applications*, Springer, 2004.

[182]. W. Zhao, R. Chellappa, and P. Phillips, "Subspace Linear Discriminant Analysis for Face Recognition," Technical Report CAR–TR–914, Center for Automation Research, University of Maryland, 1999.