The Pennsylvania State University
The Graduate School

MIXTURE MODELING FOR COMPLEX AND LARGE-SCALE DATA WITH APPLICATIONS

A Dissertation in
Computer Science and Engineering
by
Mu Qiao

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

December 2012

The dissertation of Mu Qiao was reviewed and approved* by the following:

Jia Li
Professor of Statistics and (by courtesy) Computer Science and Engineering
Dissertation Co-Adviser, Co-Chair of Committee

Daniel Kifer
Assistant Professor of Computer Science and Engineering
Co-Chair of Committee

James Ze Wang
Professor of Information Sciences and Technology
Dissertation Co-Adviser

Jesse Barlow
Professor of Computer Science and Engineering

Xiaolong (Luke) Zhang
Associate Professor of Information Sciences and Technology

Lee Coraor
Associate Professor of Computer Science and Engineering
Chair of the Graduate Program of Computer Science and Engineering

*Signatures are on file in the Graduate School.

Abstract

Mixture models have been applied in a wide range of engineering and scientific fields. In practice, data are often complex and in large scale. The data may be high dimensional, have missing values, or contain objects that are not well defined in a vector space. The flexibility of mixture models enables us to model the distribution of complex and large-scale data and estimate their probability densities. In this dissertation, we present several new mixture models that extend such application.

A two-way Gaussian mixture model (GMM) is proposed for classifying high dimensional data. This model regularizes the mixture component means by dividing variables into groups and then constraining the parameters for the variables in the same group to be identical. The grouping of the variables is not pre-determined, but rather it is optimized as part of model estimation. A dimension reduction property for a two-way mixture of distributions from a general exponential family is proved. The issue of missing values that tend to arise when the dimension is extremely high is addressed. Estimation methods for the two-way Gaussian mixture with or without missing data are derived. Experiments on several real data sets show that the parsimonious two-way mixture often outperforms a mixture model without variable grouping, and as a byproduct, significant dimension reduction is achieved.

We propose a new approach for mixture modeling based only upon pairwise distances via the concept of hypothetical local mapping (HLM). This work is motivated by increasingly commonplace applications involving complex objects that cannot be effectively represented by tractable mathematical entities. The new modeling approach consists of two steps. A distance-based clustering algorithm is applied first. Then HLM takes as input the distances between the training data and their corresponding cluster centroids and estimates the model parameters. In the special case where all the training data are taken as cluster centroids, we

obtain a distance-based kernel density. We have examined the classification performance of the mixture models on many data sets. Experimental comparisons have been made with other state-of-the-art distance-based classification methods, for instance, k-NN, variations of k-NN, and SVM based algorithms. It is found that HLM based algorithms are highly competitive in terms of classification accuracy, and in the mean time are computationally efficient during both training and testing. Furthermore, the HLM based modeling approach adapts readily to incremental learning, a valuable mechanism to achieve scalability for dynamic data arriving at a high velocity. We have developed two schemes of incremental learning and tested them on several data sets.

Driven by demands in visual analytics, we investigate a GMM with component means constrained in a pre-selected subspace, which allows us to visualize the clustering or classification structure of high dimension data in a lower dimensional subspace. We prove that the subspace containing the component means of a GMM with a common covariance matrix also contains the modes of the density and the class means. This finding motivates us to identify a subspace by applying weighted principal component analysis to the modes of a kernel density and class means. For choosing the kernel bandwidth, we acquire multiple subspaces from the kernel densities based on a sequence of bandwidths. The GMM constrained by each subspace is estimated, and the model yielding the maximum likelihood is chosen. A dimension reduction property is proved in the sense of being informative for classification or clustering. Experiments on real and simulated data sets are conducted to examine several ways of determining the subspace and to compare with the reduced rank mixture discriminant analysis (MDA). Our new method with the simple technique of spanning the subspace only by class means often outperforms the reduced rank MDA when the subspace dimension is very low, making it particularly appealing for visualization.

Finally, to bridge the computational gap between information visualization and data mining in visual analytics, we implement two parallel versions of hierarchical mode association clustering (HMAC), a previously proposed nonparametric method that groups data points into a cluster if they are associated with the same mode of a mixture-type density. Parallel HMAC runs on a cluster of compute nodes using the message passing interface (MPI) library and dramatically improves the speed of the original algorithm, making it feasible to perform clustering on large-scale data.

Table of Contents

List of Figures

List of Tables

Acknowledgments

First and foremost I would like to thank my principal advisor, Professor Jia Li, for her great guidance, motivation, enthusiasm, and support. She has not only taught me many essential research skills but also scientific attitudes, especially, creativity, visionary thinking, and the courage to solve difficult problems. Professor Li gave me thoughtful guidance and tremendous support during the tough times in my Ph.D. pursuit. Her enthusiasm and love for research is contagious. She has been a role model for me in so many ways. I am truly indebted and thankful to her.

I would like to thank my co-advisor Professor James Z. Wang. Whenever I have had questions during my Ph.D. study, he has always given me advice, shared personal stories with me, and has encouraged and inspired me. I am very grateful for his guidance, encouragement, and substantial support.

I would like to thank Professor Xiaolong Zhang. He led me into the visual analytics research field and taught me how to conduct interdisciplinary research and collaborate with people, for which I am very grateful.

I would like to thank Professor Daniel Kifer and Professor Jesse Barlow for serving on my committee, spending much time reviewing this dissertation, and providing many helpful comments and valuable suggestions.

I would like to acknowledge my debt to Professor John Yen for his guidance and support during my Ph.D. study. He led me into research at first glance. I am also thankful to Professor Damla Senturk, whose truly inspiring teaching in statistics intrigued my interest in statistical learning.

I extend my sincere thanks to my fellow Ph.D. students and friends. I want to thank Sooyoung Oh for his great help when I was a junior Ph.D. student. We had many interesting and in-depth discussions. I also want to thank Eun Yeong Ahn, Po-Chun Chen, Bi Chen, Honglu Du, Qi Fang, Liang Gou, Hyun-Woo Kim, Haibing Liu, Xingjie Liu, Huajing Li, Qinghua Li, Xin Lu, Baojun Qiu, Neela Sawant, Poonam Suryanarayan, Pucktada Treeratpituk, Anna Wu, Xin Yan, Lei Yao, Wen Yao, Mao Ye, Xiao Zhang, Yu Zhang, Kang Zhao, Yu Zhao, Ding Zhou, Shuyi Zheng, Shizhuo Zhu, and many others. Their help and collaboration have

made this journey far more interesting than it would otherwise have been.

I would like to thank my wife Jing Peng for her great love, care, and dedication throughout our life together. We have so much in common and also complement each other. She has been there for me at every step of my Ph.D. study. Her endless love and faithful support carried me through this journey. I also owe my thanks to my grandmother Huanrong Zhu and my younger brother Dai Qiao for their love, care, and support.

At last, I would like to thank my father Yuzhong Qiao and my mother Dongbing Sun for their unconditional love, unflinching sacrifice, and infinite support. They are my life models. From them, I learned integrity, honesty, gratitude, and humility. They taught me to have the courage to pursue my dreams and taught me the values that I very much cherish today.

Dedication

To my mother Dongbing Sun, my father Yuzhong Qiao, and my wife Jing Peng

Chapter 1

Introduction

Data in the real world are often very complex and in large scale. The data may have high dimensions, missing values, or contain objects that are more complicated than those in a vector space (Salton et al., 1975), for instance, sets of weighted and unordered vectors (Li and Wang, 2008). In the big data era, we frequently face problems with data of large volume, high variety, and fast velocity. Such complexity combining with the scale of data poses tremendous challenges to the design and development of new algorithms and tools. This dissertation contributes a set of new mixture models that aim to model the distribution of complex and large-scale data and estimate their probability densities, with applications in classification and clustering.

Mixture modeling has been used in various fields, for instance, to verify speakers (Reynolds et al., 2000), to classify types of limb motion (Huang et al., 2005), to predict topics of news articles (Li and Zha, 2006), and to tag online text documents (Song et al., 2008). The prominence in broad applications held by mixture models speaks for their appeals, which come from several intrinsic strengths of the generative modeling approach as well as the power of mixture modeling as a density estimation method for multivariate data.

Mixture discriminant analysis (MDA), developed by Hastie and Tibshirani (1996), has been used widely in classification. Although discriminative approaches to classification, for instance, support vector machine (Schölkopf et al., 1999), are often argued to be more favorable because they optimize the classification boundary directly, MDA, as a generative modeling method, holds multiple practical ad-

vantages including the ease of handling a large number of classes, the convenience of incorporating domain expertise, and the minimal effort required to treat new classes in an incremental learning environment. The mixture model, in particular, is inherently related to clustering or quantization if each mixture component is associated with one cluster (Celeux and Govaert, 1992; Banfield and Raftery, 1993; McLachlan and Peel, 2000). This insight was exploited by Li and Wang (2008) to construct a mixture-type density for sets of weighted and unordered vectors that form a metric but not vector space, providing additional evidence for the great flexibility of mixture modeling.

In Chapter 2, under the paradigm of MDA, we propose a two-way Gaussian mixture model (GMM) for classifying high dimensional data. This model regularizes the mixture component means by dividing variables into groups and then constraining the parameters for the variables in the same group to be identical. The grouping of the variables is not pre-determined, but rather it is optimized as part of the model estimation. A dimension reduction property for a two-way mixture of distributions from a general exponential family is proved. The issue of missing values that tend to arise when the dimension is extremely high is addressed. Estimation methods for the two-way GMM with or without missing data are derived. Experiments on several real data sets show that the parsimonious two-way mixture often outperforms a mixture model without variable grouping, and as a byproduct, significant dimension reduction is achieved.

Many data are not defined in the vector space, for instance, histogram-based descriptors, which have been widely used in image annotation (Li and Wang, 2008), shape matching (Li et al., 2000), and computer vision tasks (Yao et al., 2012). Each descriptor is essentially a set of weighted vectors, describing a discrete distribution. Some data in the real world may not be effectively represented by tractable mathematical entities, for example, the protein data in bioinformatics, the multimedia data that integrates images, texts, audios, and videos. Abstracting such data may lead to information loss and inaccuracy. Instead of modeling each data object in a mathematical format, certain distance measure may be more easily defined to capture data information by comparing their pairwise similarities. For such type of complex data, a distance-based mixture model via the concept of hypothetical local mapping (HLM) is proposed in Chapter 3. HLM takes as input the distances

between all the training data and their corresponding cluster centroids and estimates the model parameters. In the special case where all the training data are taken as cluster centroids, we obtain a distance-based kernel density. Since only pairwise distances are required for estimation, HLM is particularly appealing to model the distribution of data that are complex and cannot be easily described by a mathematical representation. Experimental results show that HLM based algorithms are highly competitive in terms of classification accuracy and computational efficiency, comparing with other state-of-the-art distance-based classification methods, for instance, k-NN, variations of k-NN, and SVM based algorithms. The HLM based modeling approach lends itself readily to the incremental learning scenario, which becomes increasingly important with the abundance of dynamic data arriving at a high velocity.

In the big data era, the explosion of information not only provides abundant resources for discovery, but also poses great challenges to human cognition, that is, how to find meaningful knowledge or patterns from data and abstract them in effective representations so that human brains can quickly absorb information and possibly identify innovative findings. Visual analytics, which integrates the sciences and technologies from data mining, information visualization, human computer interaction, and many other domains, plays a critical role in dealing with this challenge. It attempts to facilitate analytic reasoning through interactive visualization and the coupling of human and machine computational analysis.

The mixture model has its own specialties in visual analytics. First, as a clustering or classification approach, the mixture model reveals the hidden patterns of massive data. Second, it provides relatively rich geometric insight for visualization. In the mode-based clustering of data, a cluster corresponds to a "bump" or a "hill" in the probability density of a mixture model. The local maximum associated with the hill, that is, the hilltop, is referred to as the mode. A ridgeline linking two hilltops can be found to measure the separability between clusters (Li et al., 2007). It is proved that the ridgeline passes through all the critical points, such as modes, antimodes, and saddle points of the mixture density of the two hills (Ray and Lindsay, 2005). The mixture model thus provides users with heuristics about the geometric properties of data. Third, it can serve as a data reduction tool, which visualizes the data in an informative lower dimensional subspace (Hastie

and Tibshirani, 1996; Qiao and Li, 2012). Finally, the availability of fast parallel algorithms for estimating mixture models bridges the gap between real-time visualization and computationally intensive statistical modeling.

Encouraged by the favorable characteristics of mixture models for visual analytics, in Chapter 4, we investigate a GMM with component means constrained in a pre-selected subspace, which allows us to visualize the clustering or classification structure of high dimension data in a lower dimensional subspace. Applications to classification and clustering are explored. An EM-type estimation algorithm is derived. We prove that the subspace containing the component means of a GMM with a common covariance matrix also contains the modes of the density and the class means. This finding motivates us to identify a subspace by applying weighted principal component analysis to the modes of a kernel density and the class means. For choosing the kernel bandwidth, we acquire multiple subspaces from the kernel densities based on a sequence of bandwidths. The GMM constrained by each subspace is estimated, and the model yielding the maximum likelihood is chosen. A dimension reduction property is proved in the sense of being informative for classification or clustering. Experiments on real and simulated data sets are conducted to examine several ways of determining the subspace and to compare with the reduced rank mixture discriminant analysis (MDA). Our new method with the simple technique of spanning the subspace only by class means often outperforms the reduced rank MDA when the subspace dimension is very low, making it particularly appealing for visualization.

In Chapter 5, we introduce two parallel versions of hierarchical model association clustering (HMAC). HMAC is a nonparametric clustering method which groups data points into one cluster if they are associated with the same mode in a mixture-type density (Li et al., 2007). It has been applied to segment images for the analysis of color combination aesthetics (Yao et al., 2012) and to perform clustering on industry engineering design data in work-centered visual analytics to aid the search for optimal designs (Yan et tal., 2012a). Our parallel implementations of HMAC run on a cluster of compute nodes, using the message passing interface (MPI) library. Experimental results show that PHMAC significantly reduces the running time of the original algorithm, making it feasible to perform clustering on large-scale data. When used in visual analytics, the fast parallel algorithms can

render the clustering results to the visualization component within a very short time, which enables real time human and machine interaction.

Finally, we conclude and discuss future work in Chapter 6.

Chapter 2

Two-way Gaussian Mixture Models

2.1 Introduction

Mixture discriminant analysis (MDA), developed by Hastie and Tibshirani (1996), has enjoyed wide spread applications. The prominence in broad applications held by mixture models speaks for their appeals, which come from several intrinsic strengths of the generative modeling approach to classification as well as the power of mixture modeling as a density estimation method for multivariate data (Fraley and Raftery, 2002). Although discriminative approaches to classification, e.g., support vector machine (Schölkopf et al., 1999), are often argued to be more favorable because they optimize the classification boundary directly, generative modeling methods hold multiple practical advantages including the ease of handling a large number of classes, the convenience of incorporating domain expertise, and the minimal effort required to treat new classes in an incremental learning environment.

As with other approaches to classification, many research efforts on MDA revolve around the issue of high dimensionality. For the Gaussian mixture, the issue boils down to the robust estimation of the component-wise covariance matrix and mean vector. Earlier work focused more on the covariance because the maximum likelihood estimation often yields singular or nearly singular matrices when the dimension is high, causing numerical breakdown of MDA. The same issue arises for linear or quadratic discriminant analysis (LDA, QDA), less seriously than for MDA though. An easy way to tackle this problem is to use diagonal covariance matrices. Friedman (1989) developed a regularized discriminant analysis in which

the component-wise covariance matrix is shrunk towards a diagonal or a common covariance matrix across components. Banfield and Raftery (1993) decomposed the covariance matrix into parts corresponding to the volume, orientation, and shape of each component. Parsimonious mixture models were then proposed by assuming shared properties in those regards for the covariances in different components.

Recently, research efforts have been devoted to constraining the mean vectors as well. It is found that when the dimension is extremely high, for instance, larger than the sample size, regularizing the mean vector results in better classification even when the covariance structure is maintained highly parsimoniously or when covariance is not part of the estimation. For instance, Guo et al. (2006) extended the centroid shrinkage idea of Tibshirani et al. (2003) and proposed to regularize the class means under the LDA model. Some dimensions of the mean vectors are shrunk to common values so that they become irrelevant to class labels, achieving variable selection. Pan and Shen (2007) employed the L_1 norm penalty to shrink mixture component means towards the global mean so that some variables in the mean vectors are identical across components, again resulting in variable selection. Along this line of research, Wang and Zhu (2008) proposed the L_∞ norm penalty to regularize the means and select variables.

In this work, we investigate another approach to regularizing the mixture component means. Specifically, we divide the variables into groups and assume identical values for the means of variables in the same group under one component. This idea was first explored by Li and Zha (2006) for a mixture of Poisson distributions (more accurately, a product of independent Poisson distributions for multivariate data). They called such a model a two-way mixture, reflecting the observation that the mixture components induce a partition of the sample points, each usually corresponding to a row in a data matrix, while the variable groups form a partition of the columns in the matrix. Another related line of research is the simultaneous clustering or biclustering approach (Lazzeroni and Owen, 2002; Zha et al., 2001), where sample points and their variables are simultaneously clustered to improve the clustering effectiveness and cluster interpretability. Lazzeroni and Owen (2002) introduced the notion of plaid model which leads to simultaneous clustering with overlapping. Unlike two-way mixture, the simultaneous clustering approach focuses on a set of data samples and does not provide a generative

model for an arbitrary sample point, in a strict sense. Here, we study the two-way mixture of Gaussians for continuous data and derive its estimation method. The issue of missing data that tends to arise when the dimension is extremely high is addressed. Experiments are conducted on several real data sets with moderate to very high dimensions. A dimension reduction property of the two-way mixture of distributions from any exponential family is proved.

Our motivation for exploring the two-way mixture is multifold. First, in engineering applications, very differently from science where we seek a simple explanation, black box classifiers are well accepted. In scientific studies, the features (aka variables) often have natural meanings, for instance, each feature corresponds to a gene; and the purpose is to reveal the relationship between the features and some other phenomenon. Variable selection is desired because it identifies features relevant to the phenomenon. In engineering systems, the features are often defined and supplied artificially; and the purpose is to achieve good prediction performance with as much information as possible. Therefore selecting features may not be a concern, but how to combine their forces is critical. We thus focus on a parsimonious mixture model that can be more robustly estimated, but not implying the discard of any features. Moreover, estimating the two-way mixture model is computationally less intensive than selecting variables using L_1 or L_∞ norm penalty.

Second, from model estimation perspective, assuming identical means for variables in the same group is essentially to quantize the unconstrained means of the variables and replace those means by a smaller number of quantized values. Consider the following hypothetical setup. Suppose the means of k variables X_1, ..., X_k are independently sampled from a normal distribution $\mathcal{N}(0, s^2)$. Denote the means by μ_1, ..., μ_k. Suppose X_j, $j = 1$, ..., k, are independently sampled n times from $\mathcal{N}(\mu_j, \sigma^2)$, the samples denoted by $x_j^{(i)}$, $i = 1$, ..., n, $j = 1$, ..., k. Without regularization, the maximum likelihood estimation for μ_j is $\hat{\mu}_j = \sum_{i=1}^n x_j^{(i)}/n$. The total expected squared error is $E\left[\sum_{j=1}^k (\mu_j - \hat{\mu}_j)^2\right] = k\sigma^2/n$. On the other hand, if the constrained estimator $\hat{\mu} = \sum_{j=1}^k \sum_{i=1}^n x_j^{(i)}/nk$ is used for all the μ_j's, the total expected squared error is $E\left[\sum_{j=1}^k (\mu_j - \hat{\mu})^2\right] = (k-1)s^2 + \sigma^2/n$. We see that if $s^2 < \sigma^2/n$, the constrained estimator $\hat{\mu}$ yields lower total expected squared error than $\hat{\mu}_j$, $j = 1$, ..., k. The rationale for the quantization strategy is that if

we substitute $\hat{\mu}_j$'s as the true μ_j's and divide them into groups of similar values, the μ_j's in the same group are considered to be sampled from a distribution with a small s^2, and hence have a good chance of satisfying the inequality above.

The rest of this chapter is organized as follows. The two-way Gaussian mixture model is formulated in Section 2.2. We consider two cases for the component covariance matrices: diagonal for very high dimensions and unconstrained for moderately high dimensions. In Section 2.3, for the two-way mixture of distributions from any exponential family, a dimension reduction property is presented, with proof in the Appendix. The estimation algorithm and the method to treat missing data are described in Section 2.4. Experimental results with comparisons are provided in Section 2.5. Finally, we conclude and discuss future work in Section 2.6.

2.2 Two-way Gaussian Mixture Model

Let $\mathbf{X} = (X_1, X_2, ..., X_p)^t$, where p is the dimension of the data, and the class label of $\mathbf{X}$ be $Y \in \mathcal{K} = \{1, 2, ..., K\}$. A sample of $\mathbf{X}$ is denoted by $\mathbf{x} = (x_1, x_2, ..., x_p)^t$. We present the notation for a general Gaussian mixture model assumed for each class before introducing the two-way model. The joint distribution of $\mathbf{X}$ and Y under a Gaussian mixture is $f(\mathbf{X} = \mathbf{x}, Y = k) = a_k f_k(\mathbf{x}) = a_k \sum_{r=1}^{R_k} \pi_{kr} \phi(\mathbf{x}|\boldsymbol{\mu}_{kr}, \boldsymbol{\Sigma}_{kr})$, where a_k is the prior probability of class k, satisfying $0 \leq a_k \leq 1$ and $\sum_{k=1}^{K} a_k = 1$, and $f_k(\mathbf{x})$ is the within-class density for $\mathbf{X}$. R_k is the number of mixture components used to model class k, and the total number of mixture components for all the classes is $M = \sum_{k=1}^{K} R_k$. Let π_{kr} be the mixing proportions for the rth component in class k, $0 \leq \pi_{kr} \leq 1$, $\sum_{r=1}^{R_k} \pi_{kr} = 1$. $\phi(\cdot)$ denotes the pdf of a Gaussian distribution: $\boldsymbol{\mu}_{kr}$ is the mean vector for component r of class k and $\boldsymbol{\Sigma}_{kr}$ is the corresponding covariance matrix. To avoid notational complexity, we write the above mixture model equivalently as follows

$$f(\mathbf{X} = \mathbf{x}, Y = k) = \sum_{m=1}^{M} \pi_m p_m(k) \phi(\mathbf{x}|\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) \ , \tag{2.1}$$

where $1 \leq m \leq M$ is the new component label assigned in a stacked manner to all the components in all the classes. The prior probability for the mth component $\pi_m = a_k \pi_{kr}$ if m is the new label for the rth component in the kth class. Specifically,

let $\overline{R}_k = \sum_{k'=1}^{k} R_{k'}$ and $\overline{R}_0 = 0$. Then $M = \overline{R}_K$. Let the set $\mathcal{R}_k = \{\overline{R}_{k-1}+1, \overline{R}_{k-1}+2, ..., \overline{R}_k\}$ be the set of new labels assigned to the R_k mixture components of class k. The quantity $p_m(k) = 1$ if component m "belongs to" class k and 0 otherwise. That is, $p_m(k) = 1$ only for $m \in \mathcal{R}_k$, which ensures that the density of $\mathbf{X}$ within class k is a weighted sum over only the components inside class k. Moreover, denote the associated class of component m by $b(m)$. If $p_m(k) = 1$, $b(m) = k$. Then we have $a_k = \sum_{m \in \mathcal{R}_k} \pi_m$ and $\pi_{kr} = \pi_{\overline{R}_{k-1+r}}/a_k$.

2.2.1 Two-way Mixture with Diagonal Covariance

If the data dimension is very high, we adopt diagonal covariance matrix $\mathbf{\Sigma}_m = diag(\sigma^2_{m,1}, ..., \sigma^2_{m,p})$, i.e., the variables are independent within each mixture component. Model (2.1) becomes

$$f(\mathbf{X} = \mathbf{x}, Y = k) = \sum_{m=1}^{M} \pi_m p_m(k) \prod_{j=1}^{p} \phi(x_j | \mu_{m,j}, \sigma^2_{m,j}) \ . \tag{2.2}$$

In Model (2.2), the variables are in general not independent within each class as one class may contain multiple mixture components. To approximate the class conditional density, the restriction of diagonal covariance matrix on each component can be compensated by having more additive components. With diagonal covariance matrices, it is convenient to treat missing values, a particularly useful trait for applications highly prone to missing values, for instance, microarray gene expression data where more than 90% of the genes miss some measurements (Ouyang et al., 2004). We will show that the two-way Gaussian mixture model with diagonal covariance matrices can handle missing data effectively. On the other hand, for moderately high dimensional data, we will propose shortly a two-way mixture with full covariance matrices.

For Model (2.2), we need to estimate parameters $\mu_{m,j}$ and $\sigma^2_{m,j}$ for each dimension j in each mixture component m. When the dimension p is very high, sometimes $p \gg n$, we may need a more parsimonious model. We now introduce the two-way mixture model with a grouping structure imposed on the variables. In order not to confuse with the clustering structure of samples implied by the mixture components, we follow the naming convention used by Li and Zha (2006): "cluster"

refers to a variable cluster and "component" means a component in the mixture distribution. For each class k, suppose the variables are grouped into L clusters. The cluster identity of variable j in class k is denoted by $c(k,j) \in \{1,2,...,L\}$, $k = 1, ..., K$, $j = 1, ..., p$, referred to as the *cluster assignment function.* The two-way Gaussian mixture is formulated as follows:

$$f(\mathbf{X}=\mathbf{x}, Y=k) = \sum_{m=1}^{M} \pi_m p_m(k) \prod_{j=1}^{p} \phi(x_j|\mu_{m,c(b(m),j)}, \sigma^2_{m,c(b(m),j)}) \ . \tag{2.3}$$

Within each mixture component, variables belonging to the same cluster have identical parameters since the second subscripts for μ and σ^2 are given by the variable cluster assignment function. Thus, for a fixed mixture component m, only L, rather than p, μ's and σ^2's need to be estimated. Also note that $c(k,j)$ is not pre-specified, but optimized as part of model estimation. In our current study, the cluster assignment function $c(k,j)$ depends on class label k, but extension to a component specific assignment is straightforward.

2.2.2 Two-way Mixture with Full Covariance

When the data dimension is moderately high, one may suspect that diagonal covariance matrices adopted in Model (2.2) are not efficient for modeling the data and full covariance matrices can fit the data better with a substantially fewer number of components in the mixture. In order to exploit a two-way mixture as entailed in (2.3), we propose to first model the within-class density by a Gaussian mixture $f(\mathbf{X}=\mathbf{x}, Y=k) = \sum_{m=1}^{M} \pi_m p_m(k) \phi(\mathbf{x}|\boldsymbol{\mu}_m, \widetilde{\boldsymbol{\Sigma}}_k)$, where $\widetilde{\boldsymbol{\Sigma}}_k$ is an unconstrained common covariance matrix across all the components in class k. Once $\widetilde{\boldsymbol{\Sigma}}_k$ is identified, a linear transform (a "whitening" operation) can be applied to $\mathbf{X}$ so that the transformed data follow a mixture with component-wise diagonal covariance matrix, more specifically, the identity matrix $\mathbf{I}$. Assume $\widetilde{\boldsymbol{\Sigma}}_k$ is non-singular and hence positive definite, we can write $\widetilde{\boldsymbol{\Sigma}}_k = (\widetilde{\boldsymbol{\Sigma}}_k^{\frac{1}{2}})^t (\widetilde{\boldsymbol{\Sigma}}_k^{\frac{1}{2}})$, where $\widetilde{\boldsymbol{\Sigma}}_k^{\frac{1}{2}}$ is full ranked. Let $W_k = ((\widetilde{\boldsymbol{\Sigma}}_k^{\frac{1}{2}})^t)^{-1}$ and $\mathbf{Z} = W_k \mathbf{X}$. The distribution of $\mathbf{Z}$ and Y is

$$g(\mathbf{Z}=\mathbf{z}, Y=k) = \sum_{m=1}^{M} \pi_m p_m(k) \phi(\mathbf{z}|W_k\boldsymbol{\mu}_m, \mathbf{I}) \ . \tag{2.4}$$

In the light of the above model for $\mathbf{Z}$, (2.3) is a plausible parsimonious model to impose on $\mathbf{Z}$ by the idea of forming variable clusters. In fact, the covariance matrix $\mathbf{I}$ in (2.4) is not as general as the diagonal covariance matrix assumed in Model (2.3). In our study, we adopt Model (2.3) directly for $\mathbf{Z}$ instead of fixing the covariance matrix to $\mathbf{I}$, allowing more flexibility in modeling. In initialization, however, it is reasonable to set the mean of $\mathbf{Z}$ in component m as $\boldsymbol{\nu}_m = W_k \boldsymbol{\mu}_m$ and the covariance matrix $\boldsymbol{\Sigma}_m = \mathbf{I}$.

In summary, let the two-way Gaussian mixture for $\mathbf{Z}$ be

$$g(\mathbf{Z} = \mathbf{z}, Y = k) = \sum_{m=1}^{M} \pi_m p_m(k) \prod_{j=1}^{p} \phi(z_j | \nu_{m,c(b(m),j)}, \sigma^2_{m,c(b(m),j)}) \ .$$

Since $\mathbf{X} = W_k^{-1}\mathbf{Z}$, we can transform $\mathbf{Z}$ back to $\mathbf{X}$ and obtain the distribution for the original data:

$$f(\mathbf{X} = \mathbf{x}, Y = k) = \sum_{m=1}^{M} \pi_m p_m(k) \phi(\mathbf{x} | W_k^{-1}\boldsymbol{\nu}_m, (W_k^{-1})\boldsymbol{\Sigma}_m(W_k^{-1})^t), \tag{2.5}$$

where $\boldsymbol{\nu}_m = (\nu_{m,c(b(m),1)}, ..., \nu_{m,c(b(m),p)})^t$, and $\boldsymbol{\Sigma}_m = diag\left(\sigma^2_{m,c(b(m),1)}, ..., \sigma^2_{m,c(b(m),p)}\right)$.

We thus have two options when employing the two-way Gaussian mixture: (a) if the data dimension is too high for using a full covariance matrix, we assume diagonal covariance matrix as in Model (2.3); (b) if a full covariance matrix is desired, we suggest Model (2.5) which involves essentially whitening all the mixture components and then assuming Model (2.3) for the transformed data.

As a final note, to classify a sample $\mathbf{X} = \mathbf{x}$, the Bayes classification rule is used: $\widehat{y} = \text{argmax}_k f(Y = k | \mathbf{X} = \mathbf{x}) = \text{argmax}_k f(\mathbf{X} = \mathbf{x}, Y = k)$.

2.3 Dimension Reduction

In this section, we present a dimension reduction property for the two-way mixture of distributions from a general exponential family. Consider a univariate distribution from an exponential family assumed for the jth variable in $\mathbf{X}$: $\phi(x_j|\boldsymbol{\theta}) = \exp\left(\sum_{s=1}^{S} \eta_s(\boldsymbol{\theta}) T_s(x_j) - \mathbf{B}(\boldsymbol{\theta})\right) h(x_j)$. The parameter vector $\boldsymbol{\theta}$ is reparameterized as the *canonical parameter vector* $\boldsymbol{\eta}(\boldsymbol{\theta})$ and the *cumulant generating*

function $\mathbf{B}(\boldsymbol{\theta})$. $\mathbf{T}(x_j) = (T_1(x_j), ..., T_S(x_j))^t$ is the sufficient statistic vector of x_j with size S. For a two-way mixture model, variables in the same cluster within any class share parameters. We thus have the following model:

$$f(\mathbf{X} = \mathbf{x}, Y = k) \quad = \quad \sum_{m=1}^{M} \pi_m p_m(k) \prod_{j=1}^{p} \phi\left(x_j | \boldsymbol{\theta}_{m,c(b(m),j)}\right) \ . \tag{2.6}$$

Recall that $b(m)$ is the class which component m belongs to and $c(b(m), j)$ is the cluster index the jth variable belongs to. Model (2.6) implies a dimension reduction property for the classification purpose, formally stated below.

Theorem 2.3.1. *For x_j's in the lth variable cluster of class k, l = 1,..., L, k = 1, ..., K, define $\overline{\mathbf{T}}_{l,k}(\mathbf{x}) = \sum_{j:c(k,j)=l} \mathbf{T}(x_j)$, where $\mathbf{T}(x_j)$ is the sufficient statistic vector for x_j under the distribution from the exponential family. Given $\overline{\mathbf{T}}_{l,k}(\mathbf{x})$, l = 1,...,L, k = 1, ..., K, the class label Y is conditionally independent of $\mathbf{x} = (x_1, x_2, ..., x_p)^t$.*

This theorem results from the intrinsic fact about the exponential family: the size of the sufficient statistic is fixed when the sample size increases. Here, to be distinguished from the number of data points, the sample size refers to the number of variables in one cluster because within a single data point, these variables can be viewed as i.i.d. samples. Detailed proof for the theorem is provided in Appendix A.1.

In the above statement of the theorem, for notation simplicity, we assume the number of variable clusters under each class is always L. It is trivial to extend to the case where different classes may have different numbers of variable clusters. Since the size of the sufficient statistic $\mathbf{T}(x_j)$ is S, the total number of statistics needed to optimally predict class label Y is SKL. In the special case of Gaussian distribution, the size of $\mathbf{T}(x_j)$ is $S = 2$, where $T_1(x_j) = x_j$ and $T_2(x_j) = x_j^2$. In the experiment section, we will show that similar or considerably better classification performance can be achieved with $SKL \ll p$. If the way the variables are clustered is identical across different classes, i.e., $c(k, j)$ is invariant with k, the dimension sufficient for predicting class label Y is SL since $\overline{\mathbf{T}}_{l,k}$'s are identical for different k's.

2.4 Model Estimation

To estimate Model (2.3), the EM algorithms with or without missing data are derived.

Estimation without Missing Data: The parameters to be estimated include prior probabilities of the mixture components π_m, the Gaussian parameters $\mu_{m,l}$, $\sigma^2_{m,l}$, $m = 1, ..., M$, $l = 1, ..., L$, and the cluster assignment function $c(k,j) \in \{1, 2, ..., L\}$, $k = 1, ..., K$, $j = 1, ..., p$. Denote the collection of all the parameters and the cluster assignment function $c(k,j)$ at iteration t by $\psi_t : \psi_t = \{\pi_m^{(t)}, \mu_{m,l}^{(t)}, \sigma^{2(t)}_{m,l}, c^{(t)}(k,j) : m = 1, ..., M, l = 1, ..., L, k = 1, ..., K, j = 1, ..., p\}$. Let the training data be $\{(\mathbf{x}^{(i)}, y^{(i)}) : i = 1, ..., n\}$. The EM algorithm comprises the following two steps:

1. *E-step*: Compute the posterior probability, $q_{i,m}$ of each sample i belonging to component m.

$$q_{i,m} \propto \pi_m^{(t)} p_m(y^{(i)}) \prod_{j=1}^{p} \phi\left(x_j^{(i)} | \mu^{(t)}_{m,c^{(t)}(b(m),j)}, \sigma^{2(t)}_{m,c^{(t)}(b(m),j)}\right), \tag{2.7}$$

$$\text{subject to } \textstyle\sum_{m=1}^{M} q_{i,m} = 1 \, .$$

2. *M-step*: Update ψ_{t+1} by $\psi_{t+1} = \underset{\psi'}{\text{argmax}} Q(\psi'|\psi_t)$, where $Q(\psi'|\psi_t)$ is given below. Specifically, the updated parameters are given by Eqs.(2.9) $\sim$ (2.12) to be derived shortly.

$$Q(\psi'|\psi_t) = \sum_{i=1}^{n} \sum_{m=1}^{M} q_{i,m} \log\left(\pi'_m p_m(y^{(i)}) \prod_{j=1}^{p} \phi\left(x_j^{(i)} | \mu'_{m,c'(b(m),j)}, \sigma^{2'}_{m,c'(b(m),j)}\right)\right) . \tag{2.8}$$

Based on (2.8), it is easy to see that the optimal $\pi_m^{(t+1)}$, subject to $\sum_{m=1}^{M} \pi_m^{(t+1)} = 1$, are given by

$$\pi_m^{(t+1)} \propto \sum_{i=1}^{n} q_{i,m} \, , \quad m = 1, ..., M \, . \tag{2.9}$$

The optimization of $\mu_{m,l}^{(t+1)}$, $\sigma^{2(t+1)}_{m,l}$, $m = 1, ..., M$, $l = 1, ..., L$, and $c^{(t+1)}(k,j)$, $k = 1, ..., K$, $j = 1, ..., p$, requires a numerical procedure. Our approach is to

optimize the Gaussian parameters and the cluster assignment function alternately, fixing one in each turn. Let $\eta_{k,l}$ be the number of j's such that $c(k,j) = l$. In one round, $\mu_{m,l}^{(t+1)}$, $\sigma^{2}{}_{m,l}^{(t+1)}$, and $c^{(t+1)}(k,j)$ are updated by the following equations. Each maximizes $Q(\psi_{t+1}|\psi_t)$ when the others are fixed.

$$\mu_{m,l}^{(t+1)} = \frac{\sum_{i=1}^{n} q_{i,m} \sum_{j:c^{(t)}(b(m),j)=l} x_j^{(i)}}{\eta_{b(m),l} \sum_{i=1}^{n} q_{i,m}} \tag{2.10}$$

$$\sigma^{2}{}_{m,l}^{(t+1)} = \frac{\sum_{i=1}^{n} q_{i,m} \sum_{j:c^{(t)}(b(m),j)=l} (x_j^{(i)} - \mu_{m,l}^{(t+1)})^2}{\eta_{b(m),l} \sum_{i=1}^{n} q_{i,m}} \tag{2.11}$$

$$c^{(t+1)}(k,j) = \underset{l\in\{1,\ldots,L\}}{\operatorname{argmax}} \sum_{i=1}^{n} \sum_{m\in\mathcal{R}_k} q_{i,m} \left[-\frac{(x_j^{(i)} - \mu_{m,l}^{(t+1)})^2}{2\sigma^{2}{}_{m,l}^{(t+1)}} - \log|\sigma_{m,l}^{(t+1)}| \right]. \tag{2.12}$$

The optimality of Eq.(2.10) and Eq.(2.11) can be shown easily as in the derivation of the EM algorithm for a usual mixture model. Given fixed Gaussian parameters $\mu_{m,l}^{(t+1)}$ and $\sigma^{2}{}_{m,l}^{(t+1)}$, $Q(\psi_{t+1}|\psi_t)$ can be maximized by optimizing the cluster assignment function $c^{(t+1)}(k,j)$ separately for each class k and each variable j. See (Li and Zha, 2006) for the argument that applies here likewise. The optimality of $c^{(t+1)}(k,j)$ is then obvious because of the exhaustive search through all the possible values.

Eqs.(2.10)$-$(2.12) can be iterated multiple times. However, considering the computational cost of embedding this iterative procedure in the M-step, we adopt the generalized EM (GEM) algorithm (Dempster et al., 1977), which ensures that $Q(\psi_{t+1}|\psi_t) \geq Q(\psi_t|\psi_t)$ rather than solving $\max_{\psi_{t+1}} Q(\psi_{t+1}|\psi_t)$. Thus, Eqs.(2.10)$\sim$ (2.12) are applied only once. To see that $Q(\psi_{t+1}|\psi_t) \geq Q(\psi_t|\psi_t)$, let $\widetilde{\psi} = \{\pi_m^{(t+1)}$, $\mu_{m,l}^{(t+1)}, \sigma^{2}{}_{m,l}^{(t+1)}, c^{(t)}(k,j) : m = 1,...,M, l = 1,...,L, k = 1,...,K, j = 1,...,p\}$. It is straightforward to show that $Q(\psi_{t+1}|\psi_t) \geq Q(\widetilde{\psi}|\psi_t) \geq Q(\psi_t|\psi_t)$ based on the optimality of Eqs.(2.9)$\sim$ (2.11) conditioned on other parameters held fixed. The computational cost for each iteration of GEM is linear in $npML$.

To initialize the estimation algorithm, we first choose R_k, the number of mixture components for each class k. If the training sample size of each class is roughly

equal, we assign the same number of components to each class for simplicity. Otherwise, the number of components in a class is determined by its corresponding proportion in the whole training data set. Then we randomly assign each sample to a mixture component m in the given class of that sample. The posterior probability $q_{i,m}$ is set to 1 if sample i is assigned to component m and 0 otherwise. Also, each variable is randomly assigned to a variable cluster l in that class. With the initial posterior probabilities and the cluster assignment function given, an M-step is applied to obtain the initial parameters. If any mixture component or variable cluster happens to be empty according to the random assignment, we initialize $\mu_{m,l}$ and ${\sigma^2}_{m,l}$ by the global mean and variance. During the estimation, we bound the variances ${\sigma^2}_{m,l}$ away from zero using a small fraction of the global variance in order to avoid the singularity of the covariance matrix .

Estimation with Missing Data: When missing data exist, due to the diagonal covariance matrices assumed in Model (2.3), the EM algorithm requires little extra computation. The formulas for updating the parameters in the M-step bear much similarity to Eqs. (2.9) $\sim$ (2.12). The key for deriving the EM algorithm when missing data exist is to compute $Q(\psi_{t+1}|\psi_t) = E[\log f(\mathbf{v}|\psi_{t+1}) \mid \mathbf{w}, \psi_t]$, where $\mathbf{v}$ is the complete data, $\mathbf{w}$ the incomplete, and $f(\cdot)$ the density function.

When there is no real missing data, EM takes the latent component identities of the sample points as the "conceptual" missing data. When some variables actually lack measurements, the missing data as viewed by EM contain not only the conceptually missing component identities but also the physically missing values of the variables. The derivation of $Q(\psi_{t+1}|\psi_t)$ when real missing data exist is provided in Appendix A.2. We present the EM algorithm below. Introduce $\Lambda(\cdot)$ as the *missing indicator function*, that is, $\Lambda(x_j^{(i)}) = 1$ if the value of $x_j^{(i)}$ is not missing and 0 otherwise.

1. *E-step*: Compute the posterior probability, $q_{i,m}$, $i = 1, ..., n$, $m = 1, ..., M$. Subject to $\sum_{m=1}^{M} q_{i,m} = 1$,

$$q_{i,m} \propto \pi_m^{(t)} p_m(y^{(i)}) \times \prod_{j=1}^{p} \left[\Lambda(x_j^{(i)}) \phi\left(x_j^{(i)} | \mu^{(t)}_{m,c^{(t)}(b(m),j)}, {\sigma^2}^{(t)}_{m,c^{(t)}(b(m),j)}\right) + \left(1 - \Lambda(x_j^{(i)})\right)\right] \quad (2.13)$$

2. *M-step*: Update the parameters in ψ_{t+1} by the following equations.

$$\pi_m^{(t+1)} \propto \sum_{i=1}^{n} q_{i,m}, \text{ subject to } \sum_{m=1}^{M} \pi_m^{(t+1)} = 1\,, \quad m = 1, ..., M\,. \tag{2.14}$$

For each $m = 1, ..., M$, $l = 1, ..., L$, let $\tilde{x}_{j,m,l}^{(i)} = \Lambda(x_j^{(i)})x_j^{(i)} + (1 - \Lambda(x_j^{(i)}))\mu_{m,l}^{(t)}$.

$$\mu_{m,l}^{(t+1)} = \frac{\sum_{i=1}^{n} q_{i,m} \sum_{j:c^{(t)}(b(m),j)=l} \tilde{x}_{j,m,l}^{(i)}}{\eta_{b(m),l} \sum_{i=1}^{n} q_{i,m}}\,, \tag{2.15}$$

$$\sigma^{2}{}_{m,l}^{(t+1)} = \frac{\sum_{i=1}^{n} q_{i,m} \sum_{j:c^{(t)}(b(m),j)=l} (\tilde{x}_{j,m,l}^{(i)} - \mu_{m,l}^{(t+1)})^2 + (1 - \Lambda(x_j^{(i)}))\sigma^{2}{}_{m,l}^{(t)}}{\eta_{b(m),l} \sum_{i=1}^{n} q_{i,m}}\,. \tag{2.16}$$

Let $\Omega_1 = -\frac{(x_j^{(i)} - \mu_{m,l}^{(t+1)})^2}{2\sigma^{2}{}_{m,l}^{(t+1)}} - \log|\sigma_{m,l}^{(t+1)}|$, $\Omega_2 = -\frac{\left(\mu_{m,c^{(t)}(k,j)}^{(t)} - \mu_{m,l}^{(t+1)}\right)^2 + \sigma^{2}{}_{m,c^{(t)}(k,j)}^{(t)}}{2\sigma^{2}{}_{m,l}^{(t+1)}} - \log|\sigma_{m,l}^{(t+1)}|$.

$$c^{(t+1)}(k,j) = \underset{l \in \{1,...,L\}}{\operatorname{argmax}} \sum_{i=1}^{n} \sum_{m \in \mathcal{R}_k} q_{i,m}[\Lambda(x_j^{(i)})\Omega_1 + (1 - \Lambda(x_j^{(i)}))\Omega_2]\,. \tag{2.17}$$

2.5 Experiments

In this section, we present experimental results based on three data sets with moderate to very high dimensions: (1) Microarray gene expression data; (2) Text document data; (3) Imagery data. The two-way Gaussian mixture model (two-way GMM), MDA without variable clustering (MDA-n.v.c.) and Support Vector Machine (SVM) are compared for all the three data sets. Unless otherwise noted, the covariance matrices in the mixture models are diagonal because most of the data sets are of very high dimensions, e.g., $p \gg n$. To make our presentation concise, we also recall that the total number of mixture components for all the classes is always denoted by M, and the number of variable clusters in each class is denoted by L.

Microarray Gene Expression Data: We apply the two-way Gaussian mixture model to the microarray data used by Alizadeh et al. (2000). Every sample

in this data set contains the expression levels of 4026 genes. There are 96 samples divided into 9 classes. Four classes of 78 samples in total are chosen for our experiment, in particular, 42 diffuse large B-cell lymphoma (DLBCL), 16 activated blood B (ABB), 9 follicular lymphoma (FL), and 11 chronic lymphocytic leukemia (CLL). The other classes are excluded because they contain too few points. Because the sample sizes of these 4 classes are quite different, the number of mixture components used in each class is chosen according to its proportion in the training data set. We experiment with a range of values for the total number of components M. The percentage of missing values in this data set is around 5.16%. The estimation method in the case of missing data is used. We use five-fold cross validation to compute the classification accuracy.

Fig.2.1 shows the classification error rates obtained by MDA-n.v.c.. The minimum error rate 10.90% is achieved when $M = 6$. Due to the small sample size, the classification accuracy of MDA degrades rapidly when M increases. For comparison, Fig.2.1 also shows the classification error rates obtained by the two-way GMM with $L = 20$. As we can see, the two-way GMM always yields a smaller error rate than MDA-n.v.c. at any M. With $L = 20$, the two-way GMM achieves the minimum error rate 7.26% when $M = 12$. In Fig.2.1, when $M = 4$, i.e., one Gaussian component is used to model each class, MDA is essentially QDA and the two-way GMM is essentially QDA with variable clustering. The error rate achieved by QDA without variable clustering is 13.26%, while that by QDA with variable clustering is a smaller value of 9.48%.

Table 2.1: The classification error rates in percent achieved by the two-way GMM for the microarray data

Error rate (%)	$L = 5$	$L = 10$	$L = 30$	$L = 50$	$L = 70$	$L = 90$	$L = 110$	n.v.c.
$M = 4$	8.69	**7.12**	9.48	10.82	10.82	11.93	11.93	13.26
$M = 18$	**7.26**	10.02	8.60	10.82	8.46	9.48	8.46	35.30
$M = 36$	7.35	**5.83**	7.17	6.15	7.34	7.48	6.23	44.65

Table 2.1 provides the classification accuracy of two-way GMM with different values of M and L. The minimum error rate in each row is in bold font. As Table 2.1 shows, for each row, when the number of mixture components is fixed, the lowest error rate is always achieved by the two-way GMM. According to Theorem

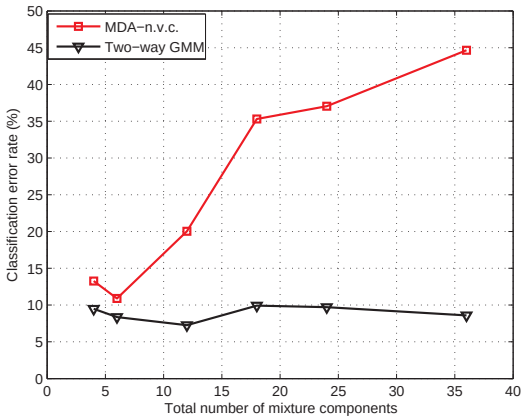


Figure 2.1: The classification error rates obtained for the microarray data using both MDA-n.v.c. and two-way GMM with $L = 20$ variable clusters. The total number of components M ranges from 4 to 36.

2.3.1, this data set can be classified with accuracy 7.26% by the two-way GMM at $M = 18$ and $L = 5$ using only 40 ($2KL = 40$) dimensions, significantly smaller than the original dimension of 4026. If homogeneous variable clustering is enforced across different classes, that is, the cluster assignment function $c(k, j)$ is invariant with class k, the classification accuracy is usually worse than the inhomogeneous clustering. Due to space limitations, we will not show the numerical results. All the results given in this section are based on inhomogeneous variable clustering.

We may use a data driven method, such as grid search and cross validation, to find the pair of M and L that gives the smallest error rate. Under some situations, the physical nature of the data may dominate the choices for M and L. For many other problems, the density of the data may be well approximated by mixture models with different values of M and L. For the purpose of classification, the mixture structure underlying the density function has no effect. It is known that discovering the true number of components assuming the distribution is precisely a mixture of Gaussian is a difficult problem and is out of the scope of this work. Effort in this direction has been made by Tibshirani and Walther (2002).

For comparison, we also apply SVM to this data set and obtain its classification accuracy with five-fold cross validation. We use the LIBSVM package (Chang and Lin, 2001) and the linear kernel with the default selection of the penalty parameter C. Missing values in the microarray data are replaced by the corresponding value

from the nearest-neighbor sample in Euclidean distance. If the corresponding value from the nearest-neighbor sample is also missing, the next nearest sample is used. The classification error rate obtained by SVM is 0.00%. Although the minimum error rate of two-way GMM listed in Table 2.1, i.e., 5.83% at $M = 36$ and $L = 10$, is larger than that of SVM, it uses only 80 ($2KL = 80$) dimensions comparing with the original dimension of 4026 used by SVM. Additionally, our focus here is not to compete with SVM, but to show that the parsimonious two-way mixture can outperform a mixture model without variable grouping.

Text Document Data: we perform experiments on the newsgroup data (Lang, 1995). In this data set, there are twenty topics, each containing about 1000 documents (email messages). We use the bow toolkit to process this data set. Specifically, the UseNet headers are stripped and stemming is applied (McCallum, 1996). A document $\mathbf{x}^{(i)}$ is represented by a word count vector $(x_1^{(i)}, x_2^{(i)}, ..., x_p^{(i)})$, where p is the vocabulary size. The number of words occurred in the whole newsgroup data is about $78,000$. In our experiment, to classify a set of topics, we pre-select words to include in the word count vectors since many words are only related to certain topics and are barely useful for the topics chosen in the data set. We use the feature selection approach described in (Li and Zha, 2006) to select the words that are of high potential for distinguishing the classes based on the variances of word counts over different classes. The feature selection in the preprocessing step is not aggressive because we still retain thousands of words. After selecting the words, we convert the word count vectors to word relative frequency vectors by normalization. Roughly half of the documents in each topic are randomly selected as training samples and the rest test samples.

We apply the two-way GMM to three different data sets, all with more than two classes. Five topics from the newsgroup data, referred to in short as, *comp.graphics, rec.sport.baseball, sci.med, sci.space, talk.politics.guns*, are used to form our first data set. Each document is represented by a vector containing the frequencies of 1000 words obtained by the feature selection approach aforementioned. In the second data set, we use the same topics as in the first one but increase the dimension of the word frequency vector to 3455. Our third data set is of dimension 5000 and contains eight topics: *comp.os.ms-windows.misc, comp.windows.x, alt.atheism, soc.religion.christian, sci.med, sci.space, sci.space, talk.politics.mideast*. In all the

three data sets, the sample size of each topic in the training data set is around 500, roughly equal to that of the test data set. We assign the same number of mixture components to each class for simplicity. Only the total number of components M is specified in the discussion.

Table 2.2 provides the classification error rates of the two-way GMM on the three data sets with different values of M and L. When M is fixed in each row, the difference between the lowest error rate achieved by the two-way GMM and the error rate of MDA-n.v.c. is also calculated. These differences are under "*diff*" in the last column of each subtable. In Table 2.2a, when $M = 5$ and 20, the lowest error rates obtained by the two-way GMM are equal to or smaller than the error rates of MDA-n.v.c.. When $M = 60$, MDA-n.v.c. gives the overall lowest error rate 8.54%, while the lowest error rate obtained by the two-way GMM is 9.27% at $L = 50$ or 110. When we increase the dimension of the word frequency vector and the number of topics to be classified, as in the second and third data sets, Table 2.2b and Table 2.2c show that the lowest error rate in each row is most of the time achieved by MDA-n.v.c.. However, the differences shown under the column of "*diff*" are always less than 1%. The performance of the two-way GMM is thus comparable to that of MDA-n.v.c., but is achieved at significantly lower dimensions. For instance, in Table 2.2c, when $M = 32$, the value under "*diff*" is 0.87% and the lowest error rate of the two-way GMM is obtained at $L = 20$. According to Theorem 2.3.1, at $L = 20$, this data set is classified using 320 ($2KL = 320$) dimensions versus the original dimension of 5000. Of particular interest is when $M = 5$ for the first and second data sets and $M = 8$ for the third data set. In those cases, a single component is assigned to each class, and hence MDA and the two-way GMM are essentially QDA with or without mean regularization. We find that for QDA, variable clustering results in lower error rates for the second data set and equal error rates for the other two.

Let us examine the two-way mixture models obtained for the two classes, *comp.os.ms-windows.misc* and *comp.windows.x*, in the third data set. Consider for example the models with $M = 32$ and $L = 30$. Fig.2.2 shows the number of words in each of the 30 word (aka variable) clusters for the two classes. These word clusters are indexed in an order of descending sizes. The sizes of these word clusters are highly uneven. In each case, the largest cluster accounts for more than

Table 2.2: The classification error rates in percent achieved by the two-way GMM for the three text document data sets

(a) Data Set 1 with five classes and dimension = 1000

Error rate (%)	$L = 10$	$L = 30$	$L = 50$	$L = 70$	$L = 90$	$L = 110$	n.v.c.	*diff*
$M = 5$	9.19	**8.95**	9.07	9.27	9.15	9.15	**8.95**	0.00
$M = 20$	12.79	9.72	9.80	**8.58**	9.15	9.39	8.99	-0.41
$M = 60$	12.06	10.04	9.27	9.80	9.39	9.27	**8.54**	0.73

(b) Data Set 2 with five classes and dimension = 3455

Error rate (%)	$L = 10$	$L = 30$	$L = 50$	$L = 70$	$L = 90$	$L = 110$	n.v.c.	*diff*
$M = 5$	7.19	**6.91**	7.07	7.07	7.11	7.15	7.15	-0.24
$M = 20$	7.88	6.99	6.79	7.88	7.84	7.11	**6.06**	0.73
$M = 60$	10.91	7.03	7.43	7.72	7.43	7.35	**6.42**	0.61

(c) Data Set 3 with eight classes and dimension = 5000

Error rate (%)	$L = 5$	$L = 10$	$L = 20$	$L = 30$	$L = 40$	$L = 50$	n.v.c.	*diff*
$M = 8$	11.41	11.06	10.96	**10.79**	10.86	10.86	**10.79**	0.00
$M = 32$	15.58	11.71	11.11	11.66	11.24	11.91	**10.24**	0.87
$M = 96$	12.79	14.26	18.23	12.29	11.79	11.09	**11.01**	0.08

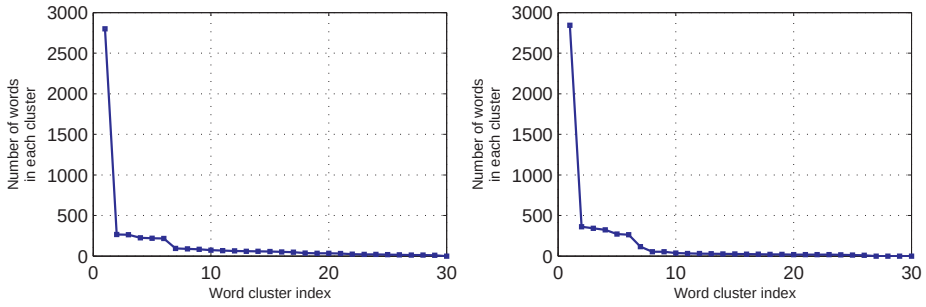


Figure 2.2: The sizes of the word clusters for *comp.os.ms-windows.misc* (left) and *comp.windows.x* (right).

half of the words. Moreover, the largest cluster contains words with nearly zero frequencies, which is consistent with the fact that for any particular topic class, a majority of the words almost never occur. They are thus treated indifferently by the model.

Classification error rates obtained by SVM for these three data sets are also reported. We use the linear kernel with different values of the penalty parameter C to do the classification. The value of C with the minimum cross validation error rate on the training data set is then selected and used for the final classi-

fication on the test data set. The SVM classification error rates on these three data sets are 7.98% (Data Set 1), 5.98% (Data Set 2) and 9.67% (Data Set 3), respectively. Comparing with the results listed in Table 2.2, SVM is only slightly better than MDA-n.v.c. and two-way GMM. However, two-way GMM achieves these error rates with a significantly smaller number of dimensions. Also, SVM is computationally more expensive and not scalable when the number of classes is large. Unlike two-way GMM, SVM does not provide a model for each class, which in some applications may be needed for descriptive purpose.

Imagery Data: The data set we used contains 1400 images each represented by a 64 dimensional feature vector. The original images contain 256×384 or 384×256 pixels. The feature vectors are generated as follows. We first divide each image into 16 even sized blocks (4×4 division). For each of the 16 blocks, the average L, U, V color components are computed. We also add the percentage of edge points in each block as a feature. The edges are detected by thresholding the intensity gradient at every pixel. In summary, every block has 4 features, 64 features in total for the entire image. These 1400 images come from 5 classes of different semantics: *mountain scenery* (300), *women* (300), *flower* (300), *city scene* (300), and *beach scene* (200), where the numbers in the parenthesis indicate the sample size of each class. Five-fold cross-validation is used to compute the classification accuracy. We use the same number of mixture components to model each class.

Table 2.3 lists the classification error rates obtained by the two-way GMM with a range of values for M and L. When M is fixed, as L increases, the error rates of the two-way GMM tend to decrease. In Table 2.3, the lowest error rate in each row is achieved by the two-way GMM. For this data set, because $2KL > 64$, dimension reduction is not obtained according to Theorem 2.3.1. However, the total number of parameters in the model is much reduced due to variable clustering, especially when M is large.

Since the dimension of the imagery data is moderate, at least comparing with the previous two data collections, we also experiment with the two-way GMM with full covariance matrices, that is, Model (2.5) in Section 2.2. Table 2.4 provides the classification error rates obtained by this model. When M is fixed, the lowest error rates are achieved by two-way GMM except at $M = 10$ and $M = 20$. Comparing

Table 2.3 with Table 2.4, the performance of the two-way GMM with full covariance matrices is slightly worse than the two-way GMM with diagonal covariance matrices. In other applications, it has also been noted that using diagonal covariance matrices often is not inferior to full covariance matrices even at moderate dimensions. One reason is that the restriction on covariance can be compensated by having more components. It is thus difficult to observe obvious improvement by relaxing the covariance.

We apply SVM with a *radial basis function* (RBF) kernel to the imagery classification problem. The penalty parameter C and the kernel parameter γ are identified by a grid search using cross validation. The final SVM error rate with five-fold cross validation is 31.00%. In Table 2.3, the minimum error rate of two-way GMM is 32.43% at $M = 40$ and $L = 36$. Similar to the previous examples, the classification accuracies of SVM and two-way GMM for the imagery data are very close. We also apply a variable selection based SVM to this classification problem since the dimension of the imagery data is moderately high. The wrapper subset evaluation method (Kohavi and John, 1997) and forward best-first search in WEKA (Hall et al., 2009) are employed to select the optimal subset of variables. In the wrapper subset evaluation method, the classification accuracy of SVM is used to measure the goodness of a particular variable subset. The final classification is obtained by applying SVM to the data with selected variables. For the SVMs involved in the variable selection scheme, the kernel function and the parameters are the same as those for the SVM without variable selection. The best subset of variables is of size 21, yielding a five-fold cross validation error rate of 34.93%. Comparing with the minimum error rates listed in Table 3 and Table 4, i.e., 32.43% ($M = 40$ and $L = 36$) and 33.21% ($M = 40$ and $L = 56$), the performance of SVM with variable selection is slightly worse than that of two-way GMM.

Table 2.3: The classification error rates in percent achieved by the two-way GMM for the imagery data

Error rate (%)	$L = 8$	$L = 12$	$L = 16$	$L = 24$	$L = 36$	$L = 48$	$L = 52$	$L = 56$	n.v.c.
$M = 5$	45.50	44.00	44.57	44.21	44.64	**43.50**	43.93	43.64	43.79
$M = 10$	40.29	37.86	36.93	35.57	35.43	35.07	35.50	**35.00**	35.57
$M = 20$	35.21	36.29	35.64	34.93	34.79	35.07	36.00	**33.93**	37.36
$M = 30$	35.43	36.07	34.86	34.64	**33.00**	34.57	34.36	34.93	34.64
$M = 40$	38.79	37.07	36.36	34.79	**32.43**	35.64	36.00	35.36	36.21
$M = 50$	37.50	35.50	33.93	34.07	**33.21**	34.14	34.93	34.93	36.50

Table 2.4: The classification error rates in percent achieved by the two-way GMM with full covariance matrices for the imagery data

Error rate (%)	$L = 8$	$L = 12$	$L = 16$	$L = 24$	$L = 36$	$L = 48$	$L = 52$	$L = 56$	n.v.c.
$M = 5$	46.57	45.86	44.21	44.57	44.57	43.93	43.93	**43.57**	43.79
$M = 10$	42.86	42.71	41.86	40.43	40.00	37.79	37.14	37.50	**35.71**
$M = 20$	42.21	43.14	39.50	38.21	36.86	37.07	36.07	35.57	**34.14**
$M = 30$	43.43	42.14	41.21	39.00	38.50	36.29	**35.79**	36.79	35.86
$M = 40$	43.64	42.00	41.50	38.43	36.29	36.07	33.64	**33.21**	33.29
$M = 50$	42.79	41.07	39.07	37.71	35.93	**33.86**	34.14	35.29	34.57

Computational Efficiency: We hereby report the running time of two-way GMM on a laptop with 2.66 GHz Intel CPU and 4.00 GB RAM. For the microarray data, when $M = 18$ and $L = 70$, it takes about 30 minutes to train the classifier on four fifths of the data and test the classifier on one fifth of the data (that is, to finish computation for one fold in a five-fold cross validation setup). For the text document data (2514 training samples, 2475 test samples, 5 classes, 3455 dimensions), when $M = 20$ and $L = 50$, it takes about 40 minutes to train and test the classifier. For the imagery data, at $M = 30$ and $L = 24$, two-way GMM with diagonal covariance matrices takes only 14 seconds to finish computation for one fold of the five-fold cross validation. The EM algorithm converges fast and the computational cost for each iteration is linear in $npML$. The longer running time required by the microarray as well as the text document data is because of the high dimensions and coding in Matlab. We expect much shorter running time if the experiments are conducted using C/C++. Although the grid search of M and L further increases the computation time, the search can be readily parallelized in a cluster computing environment.

2.6 Summary

In this work, we proposed the two-way Gaussian mixture model for classifying high dimensional data. A dimension reduction property is proved for a two-way mixture of distributions from any exponential family. Experiments conducted on multiple real data sets show that the two-way mixture model often outperforms the mixture without variable clustering. Comparing with SVM with and without variable selection, two-way mixture model achieves close or better performance. Given the importance of QDA as a fundamental classification method, we also in-

vestigated QDA with mean regularization by variable grouping, and found that the regularization results in better classification for all the data sets we experimented with.

For data sets arising out of engineering systems, variables, or features, often form natural groups according to their physical meanings. Such prior knowledge may be exploited in the future when we create variable groups in the two-way mixture. Another issue that can be explored is the component-wise whitening strategy we proposed for moderately high dimensional data when diagonal covariance matrices are considered too restrictive. In the current experiments, we did not observe gain from this strategy. It is worthy to study whether the approach can be improved by more robust estimation of covariance and whether new applications may benefit from the approach.

Chapter 3

Distance-based Mixture Modeling for Classification using Hypothetical Local Mapping

3.1 Introduction

Distance-based classifiers classify objects using only the pairwise distances between samples in the test and training data sets. Because only the pairwise distances are required in training and prediction, the specific representation of the data samples can be of greater variety and in certain cases even becomes irrelevant. This is particularly appealing to classification problems where the object cannot be described effectively by a mathematical entity that permits well-studied analytical operations, for example, vectors. The term *distance* is used in a general sense here to indicate the pairwise relationship between data samples, which can be similarity or dissimilarity, unless otherwise noted. In addition, the distance may be loosely defined, which is not necessarily a true metric. Distance-based classification is used popularly in the fields of computer vision, information retrieval, bioinformatics, etc., attracting much attention from researchers.

In this chapter, we investigate distance-based mixture modeling for classification using *hypothetical local mapping* (HLM), a mechanism originally proposed by Li and Wang (2006). HLM has been successfully applied to estimate the probabil-

ity distribution of images in a concept category, where each image is characterized by its color and texture signatures, each being a set of weighted and unordered vectors (Li and Wang, 2008). To estimate a mixture model, HLM takes as input the distances between the training image signatures and their corresponding prototype signatures. Here, by prototype, we mean the centroid of a cluster. Specifically, given a class of objects and their pairwise distances, clustering is first applied and each cluster of objects is locally mapped to a Euclidean space $\mathcal{R}^k$, preserving the pairwise distances to the maximum extent. In the mapped space, this cluster of data is modeled by a Gaussian distribution with spherical covariance and a mean vector equal to the mapped prototype. Based on the relationship between the Gaussian and Gamma distributions, the parameters of the Gaussian distribution and the dimension of the mapped space can be estimated by fitting a Gamma distribution using only distances between each data point and its corresponding prototype in the original space. The local mapping is thus bypassed, causing no additional computation, and hence called hypothetical. Finally, a mixture model is constructed to combine these clusters.

In the current work, we aim to explore the potential of HLM as a mixture modeling technique in a more general setting than what has been pursued in (Li and Wang, 2008). In the previous work, the clustering algorithm exploits the mathematical representation of an image, and is thus not pairwise distance-based. Although HLM only uses the distances after clustering, its performance coupled with distance-based clustering algorithms on more general data sets is unknown. Because the parameter fitting in HLM is computationally negligible in comparison with the pairwise distance-based clustering performed beforehand, mixture modeling by HLM is almost as fast as the clustering process itself. With the existence of some fast distance-based clustering algorithms, HLM can be substantially faster than several major pairwise distance-based classification methods. Other computational advantages of HLM will be discussed shortly. With this merit of HLM in mind, we push the consideration on computational efficiency one step further. We will propose and evaluate two incremental learning schemes of constructing mixture models by HLM. The intrinsic characteristics of HLM lend it readily to the incremental learning scenario, which becomes increasingly important with the abundance of high velocity stream data.

Comparing with discriminative approaches, such as SVM with pairwise distances modified into kernels (Hochreiter and Obermayer, 2006; Zhang et al., 2006; Chen et al., 2009), HLM inherits many practical advantages of the generative modeling approach, including the ease of handling a large number of classes, the convenience of incorporating domain expertise, and the minimal effort required to treat new classes in an incremental manner. As a generative mixture modeling approach, HLM also has profound differences from the mixture SDA (Cazzanti, 2007b). The assumption in the mixture SDA essentially puts pairwise constraints between all the classes. The estimation of the probability distribution of one class depends on the data in all the other classes, the complexity of which grows quickly with the number of classes. In contrast, HLM is along the line of mixture modeling, which establishes the density of each class separately. In addition, HLM is well linked to the method of treating pairwise distances as features. Given a set of data, if each data point is treated as a prototype, HLM will generate a mixture density using all the pairwise distances. The resulting density thus becomes a generalized kernel density.

The rest of this chapter is organized as follows. Related work is discussed in Section 3.2. We introduce HLM in Section 3.3 and the distance-based clustering in Section 3.4. The classification algorithm is described in Section 3.5. In Section 3.6, we present two HLM based incremental learning schemes. Experimental results with comparisons are provided in Section 3.7. Finally, we summarize the work and conclude in Section 3.8.

3.2 Related Work

We review distance-based classification methods in the following categories.

Nearest Neighbors: One of the most popular distance-based classification methods is k-nearest neighbor (k-NN). As a lazy learning method, k-NN does not require the training of data. Given a test sample, its distances to all the training samples are computed. The majority class of the k closest data points is assigned as the class label of the test sample. K-NN has enjoyed applications in a wide range of areas, such as image retrieval (Jacobs et al., 2000), object recognition (Belongie et al., 2002) , and photography composition classification (Yao et

al., 2012). However, k-NN is sensitive to noisy training samples, especially when the distance measure is not well defined. To mitigate this issue, various distance measures or transformations have been proposed (Pekalska et al., 2001; Simard et al., 1993; Weinshall et al., 1999). Several weighted versions of k-NN have also been proposed in (Chen et al., 2009; Cost and Salzberg, 1993; Gupta et al., 2006), which significantly improve the classification accuracy compared to standard k-NN. In addition, to speed up classification, the distances between the test sample and a set of prototypes from each class are usually used, instead of using all the training data. The prototypes are not necessarily the original data. They can be a set of edited condensing samples approximating the distribution of the original data. The strategies of prototype selection are discussed in (Lam et al., 2002; Pekalska et al., 2006; Lozano et al., 2006). If we have only one prototype from each class with the class centroid being the prototype, the scheme is also called the *nearest centroid* method.

Embed in Euclidean Space: Multi-dimensional scaling (MDS) (Young and Householder, 1938) embeds the training and test data into a lower-dimensional Euclidean space using the pairwise distances. Shepard (1962a,b) further proposed nonmetric MDS which only requires the pairwise distances in rank-order. Data can also be embedded in a pseudo-Eclidean space (Pekalska et al., 2001; Goldfarb, 1985). Standard classification methods, such as k-NN, can then be applied to the embedded data. There are several disadvantages of this approach (Cazzanti et al., 2008). First, the classification of new test data requires the re-computation of the embeddings of all the data. The embedding is computationally intensive. Second, if the distance is not a true metric, the embedding may be inappropriate. Third, the projection of data onto lower-dimensional space may cause the loss of information and thus cannot fully preserve the relationships between the original data.

Treat Distances as Features: Another distance-based classification approach is to combine the distances between a test sample and the training samples as one feature vector. Standard classifiers, such as support vector machines (SVM) and Fisher linear discriminant analysis (LDA), are then applied to the distance feature vectors (Pekalska et al., 2001; Graepel et al., 1999; Duin et al., 1999). Balcan et al. (2006) proposed a general theory of learning with distances as features, which

shows that there is a probabilistic upper bound on the classification error if almost all the samples (at least a $1 - \epsilon$ fraction) have on average closer distances to random samples in the same class than to random samples in a different class. Wang et al. (2007) further extended it to unbounded dissimilarity functions, and constructed a convex combination of simple classifiers on the distances to achieve a bounded classification error. A major limitation of this approach is that the dimensionality of the feature vector is equal to the number of training data, which can be prohibitively high. Additionally, as pointed by Chen et al. (2009), if there is large inter-class variance relative to intra-class variance, treating distances as features may not have sufficient discriminative power even though the classes are well separated.

Modify Distances into Kernels: If the pairwise distance matrix between the training samples is symmetric and positive definite, it can be treated as a kernel and used in any kernel classifiers, for instance, support vector machines. But many distance matrices do not have this property. Several methods for modifying distances into kernels are discussed in (Chen et al., 2009). Hochreiter and Obermayer (2006) proposed potential support vector machines (P-SVM) which can work with any input data matrix. However, the final kernels in these methods are all $n \times n$ matrices, which require intensive computation for large n. A hybrid approach of SVM and k-NN (Zhang et al., 2006) computes the pairwise distance matrix for the union of the test sample and its k neighbors, modifies the matrix into a kernel, and then applies standard SVM. It reduces the computational cost resulted from training on the entire data set. One problem with modifying distances into kernels is that special effort has to be made to transform the original distances between the test and training samples so that they are consistent with the modified distances in producing the kernels.

Distance-based Generative Models: This type of method was first exploited by Li and Wang (2006) for building mixture models to annotate images. They called it *hypothetical local mapping* (HLM), which requires using only distances between each data point and its corresponding prototype. Another distance-based generative model was proposed by Cazzanti and Gupta (2007a), namely, local similarity discriminant analysis (local SDA). They assume that the expectation of the similarity between a random sample x and a random class centroid μ is equal

to the mean of the similarities between the k nearest neighbors of x of the same class and the class centroid μ. Suppose the number of classes is G, this poses G^2 number of constraints. Given these constraints, each class-conditional distribution is estimated by the principle of maximum entropy. Cazzanti et al. (2008) later proposed a more generalized SDA by assuming the expectation of the descriptive statistic of a random sample x with respect to the class conditional distribution to be the average of the statistics of all the training data in each class. SDA and local SDA assume essentially a single distribution for each class. For multi-modal cases, mixture SDA was further proposed (Cazzanti, 2007b). A weighted sum of exponential components is used to estimate the class-conditional distribution. Note that in mixture SDA, G^2 number of mixture distributions have to be estimated due to a weighted version of the constraint in SDA. If the number of classes is large, the quadratic growth of the number of mixture models will present a big computational challenge.

3.3 Hypothetical Local Mapping

In this section, we introduce hypothetical local mapping (HLM) which has been used to build a generative mixture model using the distances between each data point and its corresponding prototype (Li and Wang, 2008). A prototype is a representative centroid in a cluster. If restricted to be one of the data points in the training set, a centroid is often defined as the point which has the minimum total distance to all the other points in the same cluster. The pairwise distance may be arbitrarily defined, subject to specific applications. Let us denote the distance between two objects by $\tilde{D}(\cdot, \cdot)$ and the general space where data points reside by Ω. In the following, we first introduce the estimation of data distribution in a single cluster by HLM and then show how HLM constructs a mixture model by combining these clusters.

3.3.1 Estimate the distribution of a single cluster

Let $X = (X_1, X_2, ..., X_k)^t \in \mathcal{R}^k$, where k is the dimension of the space. Suppose X is a multivariate random vector that follows a normal distribution $\mathcal{N}(\mu, \Sigma)$:

$\mu = (\mu_1, ..., \mu_k)^t$ is the mean vector and $\Sigma = \sigma^2 I$ is the covariance matrix, where I is the identity matrix. The pdf of $\mathcal{N}(\mu, \Sigma)$ is:

$$\varphi(x) = (\frac{1}{\sqrt{2\pi\sigma^2}})^k e^{-\frac{\|x-\mu\|^2}{2\sigma^2}}. \tag{3.1}$$

Denote a Gamma distribution by $(\gamma : b, s)$, where b is the scale parameter and s is the shape parameter. The pdf of $(\gamma : b, s)$ is

$$f(u) = \frac{(\frac{u}{b})^{s-1} e^{-u/b}}{b\Gamma(s)}, \quad u \geq 0$$

where $\Gamma(\cdot)$ is the Gamma function. It is known that the squared Euclidean distance between X and the mean μ, $\|X - \mu\|^2$, follows a Gamma distribution $(\gamma : \frac{k}{2}, 2\sigma^2)$.

We assume that the neighborhood around each prototype in the original space Ω, that is, the cluster associated with this prototype, can be locally mapped to $\mathcal{R}^k$ and model the mapped data in $\mathcal{R}^k$ by the Gaussian density in Equation (3.1). Expressing in terms of the Gamma distribution parameters, we have $k = 2s$ and $\sigma^2 = b/2$. To estimate the Gamma distribution parameters, we need $\| x_i - \mu \|^2$ for all data points x_i in the cluster, which are mapped from the original data objects β_i, $\beta_i \in \Omega$. Because the mapping preserves the distances, $\| x_i - \mu \|^2 = \tilde{D}(\beta_i, \alpha)$, where the prototype $\alpha \in \Omega$ is the inverse of μ. The actual mapping from Ω to $\mathcal{R}^k$ can be skipped.

Let the data point $\beta \in \Omega$ be mapped to $x \in \mathcal{R}^k$. Note again $\| x - \mu \|^2 = \tilde{D}(\beta, \alpha)$. Reformulating Equation (3.1) with the Gamma distribution parameters b and s, we obtain the density for β:

$$g(\beta) = (\frac{1}{\sqrt{\pi b}})^{2s} e^{-\frac{\tilde{D}(\beta,\alpha)}{b}}.$$

Next, we discuss the estimation of the Gamma distribution parameters b and s. Given a cluster of data, let the collection of distances between the prototype and all the other data points be $\mathbf{u} = (u_1, u_2, ..., u_N)$. Denote the mean $\bar{u} = \frac{1}{N} \sum_{i=1}^{N} u_i$.

The maximum likelihood (ML) estimator $\hat{b}$ and $\hat{s}$ are solutions of the equations:

$$\begin{cases} \log \hat{s} - \varphi(\hat{s}) = \log \left[\bar{u} / (\prod_{i=1}^{N} u_i)^{1/N} \right] \\ \hat{b} = \bar{u} / \hat{s} \end{cases}$$

where $\varphi(\cdot)$ is the di-gamma function (Evans et al., 2000):

$$\varphi(s) = \frac{d \log \Gamma(s)}{ds}, s > 0$$

The above set of equations can be solved by numerical methods (Li and Wang, 2008).

3.3.2 Estimate a Mixture Model

Suppose there are M clusters (prototypes) in Ω, with prototypes $\{\alpha_1, \alpha_2, ..., \alpha_M\}$. The overall data in Ω can be modeled by an M component mixture model. In the following, we will use cluster and component exchangeably since every mixture component is estimated using the data in one cluster. Let the prior probabilities for the components be ω_η, $\eta = 1, ..., M$, $\sum_{\eta=1}^{M} \omega_\eta = 1$. The overall mixture model for Ω is then:

$$\phi(\beta) = \sum_{\eta=1}^{M} \omega_\eta (\frac{1}{\sqrt{\pi b_\eta}})^{2s} e^{-\frac{\tilde{D}(\beta, \alpha_\eta)}{b_\eta}}. \tag{3.2}$$

The prior probability ω_η is estimated empirically by the percentage of data assigned to prototype α_η. It is assumed that the mapped spaces $\mathcal{R}^k$ of all the components have the same dimension but possibly different variances. Therefore, all the components share a common shape parameter but the scale parameter b_η varies with each component. That is, the clusters around each prototype are hypothetically mapped to the same Euclidean space $\mathcal{R}^k$, but with different spreadness. Note that the distribution of mixture model can be flexible by having more additive components. In Equation (3.2), each component distribution is similar to a Gaussian kernel in nonparametric density estimate.

Denote the index set of data points assigned to prototype j by $\mathcal{C}_j$[1], $j = 1, ..., M$. The total number of data points is $N = \sum_{j=1}^{M} |\mathcal{C}_j|$. The prior probability ω_η for

[1]Strictly speaking, we exclude from $\mathcal{C}_j$ the data point chosen as its prototype.

component η is estimated by $|\mathcal{C}_\eta|/N$. Denote the ith data point by β_i, and suppose it is assigned to cluster j. Let $u_i = \tilde{D}(\beta_i, \alpha_j)$. Suppose the mean of the distances in cluster j is $\bar{u}_j = \frac{1}{|\mathcal{C}_j|}\sum_{i\in\mathcal{C}_j} u_j$. It is shown in (Li and Wang, 2008) that the maximum likelihood estimation for s and b_j, $j = 1, ..., M$, is solved by the following equations:

$$\begin{cases} \log \hat{s} - \varphi(\hat{s}) = \log\left[\prod_{j=1}^{M} \bar{u}_j^{|\mathcal{C}_j|/N} / (\prod_{i=1}^{N} u_i)^{1/N}\right] \\ \hat{b}_j = \bar{u}_j/\hat{s}, \quad j = 1, 2, ..., M \end{cases} \tag{3.3}$$

The above equation assumes that $u_i > 0$ for every i. Although this is theoretically true, however, in practice, we may obtain singleton cluster (with only one data point) due to limited data and thus some u's are zeros. To resolve this issue, we remove all the singleton clusters which have zero distances. In addition, we may shrink $\hat{b}_j$ toward a common value, which increases the robustness of parameter estimation against clusters with small sample size. That is, modify $\hat{b}_j = \bar{u}_j/\hat{s}$ slightly to

$$\hat{b}_j = \lambda\frac{\bar{u}_j}{\hat{s}} + (1-\lambda)\frac{\bar{u}}{\hat{s}},$$

where λ is a shrinkage factor. The amount of shrinkage depends on the size of each cluster. Specifically, we set $\lambda = \frac{|\mathcal{C}_j|}{\mathcal{C}_j+1}$, which approaches 1 when the cluster size is large.

3.3.3 Estimate a Mixture Model with Weighted Distances

In some case we may have weights associated with the collection of distances $\mathbf{u} = (u_1, u_2, ..., u_N)$, that is, some distances may weigh more than others. Denote the corresponding collection of weights by $\mathbf{w} = (w_1, w_2, ..., w_N)$, where $\sum_{i=1}^{N} w_i = 1$. Let $\bar{u}_j = \frac{\sum_{i\in\mathcal{C}_j} w_i u_i}{\sum_{i\in\mathcal{C}_j} w_i}$ be the weighted mean distance for prototype j. We prove in Appendix C.1 that the maximum likelihood estimation for s and b_j, $j = 1, ..., M$, is solved by the following equations:

$$\begin{cases} \log \hat{s} - \varphi(\hat{s}) = \log\left[\prod_{j=1}^{M} \bar{u}_j^{\sum_{i\in\mathcal{C}_j} w_i} / \prod_{i=1}^{N} u_i^{w_i}\right] \\ \hat{b}_j = \bar{u}_j/\hat{s}, \quad j = 1, 2, ..., M\ . \end{cases} \tag{3.4}$$

Similar to Equation (3.3), we assume $u_i > 0$ for every i, that is, remove all the singleton clusters which have zero distances. Equations (3.3) in Section 3.3.2 are equivalent to the above equations when all the w_i's equal to $1/N$.

3.4 Distance-based Clustering

Before HLM is applied to build mixture models, we need first perform clustering on the data in each class and find the cluster centroids. Let us denote the number of data in class k by n_k. Many distance-based clustering algorithms are available. Distance-based clustering itself is a rich research topic and we do not intend to provide a comprehensive review on the related work. In this chapter, we focus on those with low computational complexity and good scalability to large datasets. We discuss the following three approaches.

Agglomerative Clustering is one of the most popular distance-based clustering methods. It starts with every data point as an individual cluster and merges similar data points together based on their pairwise distances. The merging will be stopped if the entire data has been contained in a root cluster or the desired number of clusters is achieved. Agglomerative clustering has been applied to a wide range of applications (Jain et al., 1999; Eisen et al., 1998; Beeferman and Berger, 2000). The main problem with agglomerative clustering is that it has $O(n_k^2)$ or worse computational complexity, which is prohibitive for large data sets.

Generalized k-means minimizes the total within cluster distance, in the same spirit as k-means, but only uses pairwise distances. Several data points are randomly selected as the initial set of prototypes. The assignment of the remaining data points to their closest prototypes yields a partition of the data set. For all the data points in the same group, a new prototype is updated as the one with the minimum total distance to all the other data points in the group. Then the data points are assigned again to the new prototypes. This process continues until a pre-determined maximum number of iterations is reached or the prototype in each cluster has converged, resulting in a fixed partition. Suppose the number of data points in cluster k is l_k, the total computational cost for generalized k-means is $O(\sum_{k=1}^{K} l_k^2 + Kl_k))$. Since $l_k > K$ in general, its computational cost is therefore $O(\sum_{k=1}^{K} l_k^2)$, whereas agglomerative clustering has $O(n_k^2)$ or worse complexity.

When the data set has large samples and the sizes of partitioned subsets are relatively balanced, generalized k-means reduces the computational cost considerably. Generalized k-means is also referred to as k-medoids (Hastie et al., 2001). It can be regarded as a heuristic method to the well-known p-median problem in the operation research community (Maranzana, 1964). The goal of p-median is to locate p facilities in order to minimize the total transportation cost between the facilities and n demand points. P-median is a combinatorial optimization problem, which is proved to be NP-hard (Cornuejols et al., 1977). P-median naturally arises in clustering analysis (Rao, 1971; Mulvey and Crowder, 1971), when the p "facilities" are selected to be the most representative data points. Suppose the set of prototypes is $\{\mu_1, ..., \mu_k\}$ and the assignment of data point x_i, $i = 1$, ..., N, to cluster j is $C(i) = \text{argmin}_{1 \leq j \leq k} d_{i\mu_j}$. We abuse the notation k slightly here to mean the number of prototypes. Denote the index set of data assigned to prototype j by $\mathcal{C}_j$. The total within cluster distance is defined as $\sum_{j=1}^{k} \sum_{i \in \mathcal{C}_j} d_{i\mu_j}$. P-median is to minimize this distance by locating appropriate centroids, that is,

$$\min_{C, \{\mu_j\}_1^k} \sum_{j=1}^{k} \sum_{i \in \mathcal{C}_j} d_{i\mu_j}. \tag{3.5}$$

As a heuristic method of p-median problem, generalized k-means essentially finds successively single facilities of k^2 subsets of demand points and then update the subsets before repeating the process.

Vertex Substitution Heuristic (VSH) was first proposed by Teitz and Bart (1968) as another heuristic approach to p-median problem. Similar to the generalized k-means, it first randomly selects several data points as the initial set of prototypes and assigns the remaining data points to their closest prototype. Exchange each prototype with a data point that is currently not a prototype or not previously tried. Select the exchange that results in the largest reduction in Equation (3.5). This process is repeated until no further reduction can be found. We show the details in Algorithm (1). In practice, it is found that VSH shows more stable performance than generalized k-means (Teitz and Bart, 1968). The worst case complexity of VSH is $O(n_k^2)$ when applied to the data in class k.

[2] $k = p$ in this scenario.

Algorithm 1: Vertex Substitution Heuristic (VSH)

Input: Pair-wise distances between N data points; the number of prototypes k
Output: An estimate of the set of prototypes that solves Equation (3.5)
Set $t = 1$;
Randomly select k data points as the initial set of prototypes V_t;
begin
  foreach *data point* $v_i \notin V_t$ **do**
    Compute $C(i) = \text{argmin}_{1 \le j \le k} d_{i\mu_j}$; `// assign to the nearest prototype`
  end
  Compute the total within cluster distance $r_t = \sum_{j=1}^{k} \sum_{C(i)=j} d_{i\mu_j}$;
  foreach $v_b \notin \bigcup_{i=1,\ldots,t} V_i$ *and not visited* **do**
    foreach $v_j \in V_t$ **do**
      Exchange v_b and v_j and compute the new total within cluster distance r'_t;
      Compute $\Delta_{bj} = r'_t - r_t$;
    end
    Find the data point $v_{j'}$ satisfying
     $\Delta_{bj'} < 0$ and $j' = \text{argmin}_{1 \le j \le k} \Delta_{bj}$; `// has the largest distance reduction`
    if *there exists such a data point* $v_{j'}$ **then**
      Exchange v_b and $v_{j'}$ and mark the new prototype set as V_{t+1};
      Compute r_{t+1};
    else
      $V_{t+1} = V_t$; `// retain the previous prototype set`
    end
    Mark v_b as visited;
    $t = t + 1$;
  end
  if *no more reduction in* r_t **then**
    Output V_t;
    Terminate;
  else
    Reset $t = 1$;
    Mark all the data points as unvisited. Go back to the step **begin** and repeat the above procedures;
  end
end

3.5 The Algorithm

In this section, we present two approaches to classification based on HLM. For the first approach, clustering is applied before forming a mixture density via HLM.

For the second approach, referred to as HLM (kernel), a mixture density is formed without performing clustering, but instead in the fashion of a kernel density estimate. We summarize the steps of HLM with clustering first:

1. Perform clustering on the data in each class k using distance-based clustering methods and identify the cluster centroids. Obtain the distances between each data point and its corresponding prototype.

2. Suppose the total number of prototypes for all the classes is $\bar{M} = \sum_k M_k$, where M_k is the number of prototypes in class k, $k = 1, 2, ..., K$. Let the indices of the prototypes in class k be $\mathcal{F}_k = \left\{\bar{M}_{k-1} + 1, \bar{M}_{k-1} + 2, ..., \bar{M}_{k-1} + M_k\right\}$, where $\bar{M}_{k-1} = M_1 + M_2 + \cdots + M_{k-1}$ for $k > 1$, and $\bar{M}_0 = 0$. Denote the set of points assigned to component η, $\eta = 1$, ..., $\bar{M}$, by $\mathcal{C}_\eta$.

3. Estimate a mixture model $\mathcal{M}_k$ for each class k:

$$\phi(\beta|\mathcal{M}_k) = \sum_{\eta \in \mathcal{F}_k} \omega_\eta (\frac{1}{\sqrt{\pi b_\eta}})^{2s} e^{-\frac{\tilde{D}(\beta, \alpha_\eta)}{b_\eta}}. \tag{3.6}$$

   where b_η is the scale parameter for component η and s is the common shape parameter shared by all the components in all the classes. That is, we assume that the cluster of data around each prototype in the data set are hypothetically mapped to a Euclidean space of the same dimension, but with possibly different spreadness. Note that it is also straightforward to make the shape parameter s vary with the class, that is, to have s_k. The only difference between these two is that distances from all the classes are collected and input to Equations (3.3) for the former while only distances within a class are used in the estimation for the latter. The prior probability of each component ω_η in Equation(3.6) is estimated empirically by $|\mathcal{C}_\eta| / \sum_{\eta' \in \mathcal{F}_k} |\mathcal{C}_{\eta'}|, \eta \in \mathcal{F}_k$. Also, estimate the prior probabilities of each class, π_k, $k = 1$, ..., K, by their empirical frequencies.

4. To classify a test data point t, we compute the posterior probability of t

belonging to each class k:

$$p_k(t) = \frac{\pi_k \phi(t|\mathcal{M}_k)}{\sum_{l=1}^{K} \pi_k \phi(t|\mathcal{M}_l)}, \quad k = 1, 2, ..., K \ . \tag{3.7}$$

The class label of t is then $\text{argmax}_{1 \le k \le K} \, p_k(t)$.

The role of clustering conducted before mixture modeling is two-fold: as a smoothing mechanism to suppress outliers and as a data reduction mechanism to save computation. It is found through experiments that the first benefit is not always true. On the contrary, treating all the original data as centroids often outperforms a much reduced set of cluster centroids, albeit at the cost of more computation during testing. This approach is particularly appealing when the number of data points in a class is very small, while clustering in this case is quite likely to lose valuable information. We thus develop a kernel version of HLM based mixture modeling without clustering, denoted by HLM (kernel). Specifically, each training data point is treated as a cluster centroid in the corresponding class. Suppose the training data points are $\{\beta_1, \beta_2, ..., \beta_N\}$. The number of data points in class k is n_k, $\sum_{k=1}^{K} n_k = N$. Without loss of generality, assume the indices of the data in class k be $\{\beta_{\bar{n}_k+1}, ..., \beta_{\bar{n}_k+n_k}\}$, where $\bar{n}_k = \sum_{k'=1}^{k-1} n_{k'}$, for $k > 1$, and $\bar{n}_1 = 0$. We form a nonparametric mixture model for class k:

$$\phi(\beta|\mathcal{M}_k) = \sum_{i=\bar{n}_k+1}^{\bar{n}_k+n_k} (\frac{1}{n_k})(\frac{1}{\sqrt{\pi b}})^{2s} e^{-\frac{\tilde{D}(\beta,\beta_i)}{b}} \ .$$

We need to decide an appropriate scale parameter b and shape parameter s, which are common across the clusters in all the classes. A data-driven approach is adopted. We first form pseudo clusters by grouping each data point and its nearest neighbor in the same class. All these "tiny" clusters, each containing one data point besides the cluster centroid, are input to the estimation process described in 3.3.2 to determine b and s. In practice, we find that this approach to estimate the parameters usually yields good performance.

3.6 HLM Incremental Learning

Data streams are common in the real world, for instance, network traffic, financial transactions, web search, and social media feeds. They are ordered sequences of data records that often arrive in batches with fast velocity, or in burst. It is thus desirable to have models that can exploit immediately the available training data and adapt efficiently with newly arrived data. Another motivating scenario for learning models incrementally is when the entire data cannot be loaded into memory at once. In this section, we will show that our proposed mixture models based on HLM can be estimated conveniently in an incremental learning setup. Two incremental learning schemes are proposed. The first scheme is more efficient in computation, while the second scheme attempts to achieve better clustering by combining old and new data in clustering.

3.6.1 Scheme I

In Section 3.3.2, the maximum likelihood estimation for model parameters is solved by Equations (3.3). Note that, for each cluster j, only the size of the cluster $|\mathcal{C}_j|$, the arithmetic mean of the distances to the cluster prototype $\bar{u}_j$, and the geometric mean of all the distances $\bar{g} = (\prod_{i=1}^{N} u_i)^{1/N}$, are needed in the estimation. Therefore, in an incremental learning setting, data up to the current batch can be discarded after those statistics are stored. Given a new batch of data, we first perform clustering and then obtain these statistics for each cluster. The new statistics and those from previous data are then pooled together to solve the parameters by Equation (3.3). Note that $\bar{g}$ will change since the number of data N increases when a new batch of data arrives. Suppose the number of data that have arrived so far is N_1 and the number of data in a new batch is N_2. Let $N = N_1 + N_2$. Suppose we have stored $\bar{g_1} = (\prod_{i=1}^{N_1} u_i)^{1/N_1}$. When a new batch arrives, the geometric mean of all the distances $\bar{g}$ should be updated by $\bar{g} = \bar{g_1}^{N_1/N}(\prod_{i=N_1+1}^{N} u_i)^{1/N}$.

In this scheme, the clustering is performed separately on each batch of data. Clusters (or related statistics) from the currently available data and the new batch are pooled together to estimate a new HLM. This is motivated by the additive nature of the mixture models. The distribution of data can be better approximated by having more additive components.

3.6.2 Scheme II

A weighted clustering is performed on randomly sampled points from the data up to the current batch and all the data in the new batch. We randomly sample a small number of data points in each cluster obtained previously and assign them relatively large weights. Smaller weights are given to the data points in the new batch because they are not subject to sampling. Specifically, in the current data, for a cluster j, if $|\mathcal{C}_j|$ is larger than a pre-determined threshold, denoted by v, we randomly select v samples as well as the cluster centroid from that cluster. Otherwise, for smaller $|\mathcal{C}_j|$, all the samples from that cluster will be selected. In practice, v can be a very small number. Denote the weight of the ith sampled data point by w_i. The weight of a randomly selected sample i' from cluster j is updated by

$$w_{i'} = \begin{cases} \sqrt{\frac{|\mathcal{C}_j|+1}{v+1}} \text{ , if } |\mathcal{C}_j| > v \\ 1.0 \text{ , if } |\mathcal{C}_j| \leq v \text{ .} \end{cases}$$

The above function will weigh heavier individual points in down sampled clusters with $|\mathcal{C}_j| > v$. On the other hand, the weight is dampened from a linear proportion in order to lower the influence of "old" data collectively and to reduce the effect of descreased dispersion in sampled data. When a new batch of data arrives, we set the weight of each new data point to 1.0. A weighted clustering algorithm is then applied to the combination of the sampled data and the data from the new batch. The three distance-based clustering algorithms introduced in Section 3.4 can be used to perform clustering by taking a weighted distance matrix as input. Specifically, let the distance matrix D be the original symmetric matrix of $d_{i,j}$ between all pairs of data points i and j and H be the same order diagonal matrix with weights on the diagonal. The weighted distance matrix R is defined by $R = HD$, which is no longer generally symmetric. The vertex substitution heuristic (VSH) algorithm working on weighted distance matrix is introduced in (Teitz and Bart, 1968).

After the cluster centroids are obtained, we re-assign to the nearest centroids the data points that are neither in the new batch nor among the sampled data. In this way, all the data that have arrived so far are partitioned into the new clusters,

although many do not participate in generating the centroids. The new clustering result is used to estimate the parameters. This process is repeated at the arrival of every new batch. Note that because the old cluster centroids have been assigned a relatively large weight, they are more likely to become new centroids again. In practice, we gradually increase the total number of clusters when new data arrive to increase the odds of new data becoming cluster centroids.

3.7 Experiments

In this section, we compare HLM with several other distance-based classification methods on various datasets. The classification methods are described briefly below:

1. SVM: modify the $n \times n$ symmetric distance matrix into a kernel (positive semidefinite) by spectrum clip, which clips all the negative eigenvalues to zero (Chen et al., 2009); and use a linear or Gaussian RBF kernel on similarity feature vectors.

2. Potential-SVM: a more generalized SVM working with any given $n \times n$ distance matrix, which is not necessarily square or positive definite (Hochreiter and Obermayer, 2006).

3. SVM-KNN: apply standard SVM to classification using a kernel modified from the pairwise distance matrix between the union of a test sample and its k nearest neighbors.

4. Similarity Discriminant Analysis (SDA): a set of generative classification methods based on similarities (or distances) are considered, that is, basic SDA, local SDA, and mixture SDA, where each class-conditional distribution is estimated by maximum entropy under a constraint. Both local SDA and mixture SDA aim to reduce the model bias issue of basic SDA. However, the computation of mixture SDA is very intensive, which may be infeasible for a large number of classes. Since local SDA consistently performs well in practice and is comparable to the other two (Chen et al., 2009), we select local SDA as the representative of this particular set of methods for comparison.

5. k-NN: a test sample is assigned to the class most common among its k nearest neighbors.

6. Weighted k-NN: two weight design goals are considered, that is, affinity (assign larger weights to the data points that are closer to the test sample) and diversity (assign smaller weights to highly similar data points). Specifically, three different weight assignment approaches are tested: affinity weights, kernel ridge interpolation weights (KRI), and kernel ridge regression weights (KRR) (Chen et al., 2009). KRI-KNN and KRR-KNN consider both affinity and diversity.

7. HLM (vsh, gknn, agg, kernel): the proposed mixture model using HLM, with the clustering results obtained by VSH, generalized k-means, or agglomerative clustering, or without clustering.

3.7.1 Data Sets

The information of all the tested data sets are summarized in Table 3.1. The column entitled "Symmetric" indicates whether the distance is symmetric, and the column entitled "Vector" indicates whether the data object is a vector. For details of the first eight data sets, we refer interested readers to (Chen et al., 2009). We add four more new data sets:

Imagery data has 1400 images that come from five classes of different semantics: mountain scenery (300), women (300), flower (300), city scene (300), and beach scene (200), where the numbers in the parentheses indicate the sample size of each class. Each image is represented by a 64 dimensional feature vector. The distance between images is the Euclidean distance.

Color signature has 600 images each represented by its color signature, i.e., a set of weighted vectors (or discrete distribution). 100 images are selected from each class of the above *imagery data*, with another 100 images from a new class of semantics "man-made items". In total, it has six classes. To form the color signature of an image, we first convert the RGB color components of each pixel to the LUV color components and then apply k-means on the 3-D color vectors at all the pixels. The number of clusters in k-means is determined dynamically by thresholding the average within cluster distance. An image segmentation is

obtained after arranging the cluster labels of the pixels into an image according to the pixel positions. We refer to the collection of pixels mapped to the same cluster as a region. For each region, its average color vector and the percentage of pixels it contains with respect to the whole image are computed. The color signature is thus formulated as a set of weighted vectors $(v^{(1)}, p^{(1)}), (v^{(2)}, p^{(2)}), ..., (v^{(m)}, p^{(m)})$, where $v^{(j)}$ is the mean color vector, $p^{(j)}$ is the associated probability, and m is the number of regions. The mean value of m is equal to 9.98 for this data set. The distance between images is the Mallows distance (Mallows, 1972).

Photography composition is used in (Yao et al., 2012) as the benchmark data for composition classification. There are 150 photos that are equally divided into three classes: horizontal, vertical, and centered. Each photo is represented by a set of spatial relational vectors (SRV), which quantitatively characterizes its spatial layout. Similar to the *color signature* data, each SRV is also a set of weighted vectors. We refer interested readers to (Yao et al., 2012) for details of SRV. The distance between photos is the integrated region matching (IRM) distance (Li et al., 2000), a greedy variant of Mallows distance.

Sonar data is from the UCI machine learning repository, with 208 samples divided into two classes (111, 97). Each sample has a 60 dimensional feature vector. The distance between samples is the Euclidean distance.

Table 3.1: Summary of Data Sets

Name	# data	# classes	Symmetric	Vector	Distance Type
Amazon-47	204	47	No	No	Percentage
Aural Sonar	100	2	Yes	No	Human judgment
Caltech-101	8677	101	Yes	Yes	Kernel
Face Rec	945	139	Yes	No	Cosine similarity
Mirex07	3090	10	Yes	No	Human judgment
Patrol	241	8	No	No	Frequency
Voting	435	2	Yes	Yes	Value difference metric
Protein	213	4	Yes	No	Sequence-alignment similarity
Color Signature	600	6	Yes	No	Mallows distance
Photo Composition	150	3	Yes	No	IRM distance
Imagery	1400	5	Yes	Yes	Euclidean distance
Sonar	208	2	Yes	Yes	Euclidean distance

Since HLM uses distances as input, for some datasets with similarities, we need to convert them into dissimilarities, that is, distances. Specifically, if the similarity s has an obvious upper bound u, the corresponding distance is defined as $d = u - s$, otherwise, $d = 1/s$. On the other hand, for the algorithms taking similarities as

input, we also need to convert distances into similarities. In Table 3.1, we have three different distance metrics, Mallows, IRM and Euclidean distances. Since distances have no upper bound values, the corresponding similarities are defined as their inverse. We use an appropriate upper bound of all the similarities as the self-similarity of a data point. In addition, if the distance between two data points x and y is not symmetric, for example, the data in Amazon-47 and Patrol, we use the symmetrized distance $(d(x, y) + d(y, x))/2$.

3.7.2 Experimental Setup and Details

For each data set, we randomly select 20% as test and the remaining 80% as training. The classifier parameters, such as the penalty parameter C and the RBF parameter γ for SVM, weight λ for KRI k-NN and KRR k-NN, the neighborhood size k for k-NN and local SDA, and the number of components in HLM, are all selected by 10-fold cross validation on the training set. The trained model is then applied to classify the held out test data. We repeat this process for 20 random partitions of training and test. The"one-versus-one" scheme (Hsu and Lin, 2002) is used in the multi-classification of SVM. We assume equal number of components in each class for HLM and the number of components is selected from $\{1, 2, 3, .., 10, 12, 14, 16\}$ by cross validation. The range of each parameter in cross validation for the classifiers that are tested in (Chen et al., 2009) is kept the same as the range used in that paper.

Note that in HLM, we have $2s = k$, where s is the shape parameter and k is the dimension of the mapped hypothetical space. Since the dimension of space, k, should be an integer, we adjust the ML estimation $\widehat{s}$ in Equations (3.3) to be $s^* = \lfloor 2\widehat{s} + 0.5 \rfloor$. In addition, we exclude the clusters containing fewer than three data points in the estimation of model parameters. We test three different distance-based clustering methods introduced in Section 3.4 to obtain the cluster centroids for HLM. The Ward's method (Ward, 1963) is used in agglomerative clustering. Generalized k-means algorithm is initialized by the k-center algorithm which is also based on pairwise distances. For VSH, we select the clustering which yields the minimum total within cluster distance with 20 random initializations.

3.7.3 Classification Results

The classification error rates of all the tested methods on the twelve data sets are reported in this section. For each method, we show its mean error rate and standard deviation across 20 random partitions of training and test.

Because the Amazon-47, Face Rec and Patrol data sets have very small number of training data within each class, performing clustering in each class is not meaningful. Among the HLM based algorithms, we thus only applied HLM (kernel) to these data sets. For some special data sets such as Patrol, the distances between data points only take three values: 0.0, 0.5, and 1.0. The distance between one data point and its nearest neighbor is often 0.0, causing numerical issues for the parameter estimation in HLM. Therefore, for Patrol, we manually select the common shape and scale parameters from the range of estimated values based upon HLM (vsh). HLM (kernel) is similar to k-NN in the sense that both require distances to all the training data to classify a test point. In HLM (kernel), every training data point contributes smoothly to the decision function, while in k-NN, only the closest k neighbors contribute. Note that local SDA is reduced to k-NN if there is not enough data to fit distributions over the distances (Chen et al., 2009).

We experimented with fourteen classification algorithms. These algorithms fall into several categories: k-NN and its variations, SVM based, local SDA, and HLM based. For clarity, we present results for five representative algorithms in Table 3.2. We choose one algorithm from each of the three categories in existing work, namely, k-NN, local SDA, and SVM-similarities as features (rbf). These algorithms are well known and relatively basic among their respective categories. In addition, the more complicated versions are not evidently stronger. For the newly developed HLM based algorithms, we show results of HLM (vsh) and HLM (kernel) in this table. For completeness, we report the results of the other nine algorithms in Table 3.3. For the first eight data sets listed in Table 3.1, the classification results of all the algorithms other than the HLM based are taken directly from the results in (Chen et al., 2009).

Based on Table 3.2, if we compare the generative modeling approaches, local SDA, HLM (vsh), and HLM (kernel), we see that HLM (kernel) performs best on 7 data sets, local SDA performs best on 4 data sets, and HLM (vsh) performs best on 1 data set. Taking also into consideration k-NN and SVM (rbf), HLM (kernel)

performs best among the five algorithms on 6 data sets, SVM (rbf) performs best on 4 data sets, and k-NN performs best on 1 data set (tied best with HLM). Clearly, HLM (kernel) ranks on the top most frequently. The performance of SVM (rbf) deviates remarkably from the other algorithms on several data sets. For the Protein data, it yields a significantly lower error rate than the others, 2.67% versus others ranging from 17.44% to 29.88%. On the other hand, it performs much worse than the others on Amazon-47 and Patrol. For Amazon-47, its error rate is 75.98% while the others range from 15.61% to 16.95%; for Patrol, its error rate is 40.73% while the others are tightly around 11.5%. Similar performance on these three data sets, either very good or very poor, is also observed for SVM (clip), SVM (linear), and P-SVM, as shown in Table 3.3. In a sense, the SVM-based algorithms are more volatile. Based on the results in Table 3.3, we can see that among the three HLM based algorithms, HLM (vsh) consistently works well while HLM (gknn) exhibits more variation in performance. HLM (agg) is also relatively stable in performances but is more expensive in computation.

Computational Complexity and Running Time Recall that the total number of data is $N = \sum_k n_k$, where n_k is the number of data in class k, and the total number of clusters (aka components) for all the classes is $\bar{M} = \sum_k M_k$, where M_k is the number of components in class k. The worst scenario complexity of the three clustering algorithms (vsh, gknn, agg) is $\sum_k n_k^2$. After distances between all the data points and their corresponding prototypes are obtained, statistics required in Equations (3.3) can be computed in $\mathrm{O}(N)$. We solve Equations (3.3), numerically, through binary search over a fixed range, which can be finished in $\mathrm{O}(c\bar{M})$, where c is a constant time to solve a single equation. Therefore, the total complexity of estimating mixture models for all the classes is $O(N + c\bar{M})$, given the clustering results are available. In practice, $N > \bar{M}$. So the model estimation complexity is linear in the total number of data N. The main computational cost is thus on clustering. In practice, we find that VSH runs very fast and returns good clustering results. Comparing with local SDA and SVM based classifiers, HLM (vsh) has significantly shorter running time, making it very attractive for large scale computation. The two parameters of SVM (rbf), i.e., the penalty parameter C and the RBF parameter γ, are selected by grid search using cross validation, which is extremely intensive in computation. By contrast, HLM (vsh) has only

one parameter, the number of clusters in a class, to choose by cross validation.

HLM (kernel) has exhibited highly competitive classification performance on most data sets. It is similar to k-NN in that the computation during classification is mostly on getting the distances between a test point and every training point. During training, k-NN is not totally free of computation because cross validation is used to choose k, while HLM (kernel) only estimates two parameters by an extremely fast algorithm after one round of 1-NN is performed. However, comparing with HLM (vsh) which exploits a possibly much smaller set of cluster centroids than the original data, HLM (kernel) is expensive during classification for large data sets. For such data sets, a subset of representative data points may be sampled from each class and used as training data in HLM (kernel). This approach can be further investigated in the future work.

Table 3.4 lists the running time of k-NN, local SDA, SVM (rbf), HLM (vsh), and HLM (kernel) on several exemplary data sets. The experiments run on a 2.66 GHz Intel CPU. HLM based algorithms are implemented in C, local SDA is in Matlab[3], and the remaining algorithms are in C++[4]. The value listed in the table is the average running time for one random partition of training and test data. We conduct 20 random partitions for every data set, and within each partition, training and testing are performed. As Table 3.4 shows, HLM (vsh) and HLM (kernel) run significantly faster than local SDA and SVM (rbf). For most data sets, SVM (rbf) has the longest running time while k-NN has the shortest. The running time of HLM (kernel) is close to that of k-NN, though slightly higher on some data sets. This is mainly due to the cost of choosing appropriate scale and shape parameters in HLM (kernel). For local SDA, SVM (rbf), and HLM (vsh), the classification of test data is very fast after the model has been trained. The main computational cost is therefore on training. As aforementioned, both HLM (kernel) and k-NN are expensive for classifying test data if the training data set is large. For example, on the Mirex data set, the average testing time of HLM (kernel) across 20 random partitions is 295 ms while HLM (vsh) takes only 29 ms, about ten times faster.

[3]http://staff.washington.edu/lucagc/software.html. The Matlab Executable (MEX) files are available for local SDA, which run faster but require customized installations on different platforms.

[4]http://ee.washington.edu/research/guptalab/similaritylearning/simMLL-linux.tgz.

Table 3.2: Classification error rates of distance-based classifiers (I)

Classifier	Amazon-47	Aural Sonar	Caltech-101	Color-signature
k-NN	16.95 (4.85)	17.00 (7.65)	41.55 (0.95)	32.54 (3.26)
Local SDA	16.83 (5.11)	17.75 (7.66)	41.99 (0.52)	35.71 (2.67)
SVM-similarities as features (rbf)	75.98 (7.33)	14.25 (7.46)	**38.16 (0.75)**	36.58 (3.54)
HLM (vsh)	NA	16.00 (5.39)	48.52 (1.08)	36.42 (3.42)
HLM (kernel)	**15.61 (5.37)**	**13.75 (6.50)**	40.18 (1.00)	**31.54 (3.96)**
Classifier	Face Rec	Imagery	Mirex	Patrol
k-NN	4.23 (1.43)	**36.64 (3.04)**	61.21 (1.97)	11.88 (4.42)
Local SDA	4.55 (1.67)	42.87 (2.52)	60.94 (1.94)	11.77 (4.62)
SVM-similarities as features (rbf)	3.92 (1.29)	47.16 (2.38)	**55.72 (2.06)**	40.73 (5.95)
HLM (vsh)	NA	44.73 (2.71)	61.62 (1.92)	NA
HLM (kernel)	**3.81 (1.36)**	**36.64 (3.13)**	69.42 (2.33)	**11.46 (4.09)**
Classifier	Protein	Photo Composition	Sonar	Voting
k-NN	29.88 (9.96)	25.00 (6.37)	20.24 (6.37)	5.80 (1.83)
Local SDA	17.44 (6.52)	22.67 (8.92)	**20.00 (6.79)**	6.38 (2.07)
SVM-similarities as features (rbf)	**2.67 (2.97)**	**19.67 (7.14)**	21.31 (5.71)	5.52 (1.77)
HLM (vsh)	23.49 (10.55)	23.50 (7.11)	24.40 (4.69)	**4.89 (2.12)**
HLM (kernel)	29.07 (7.52)	23.17 (8.66)	23.81 (5.05)	6.09 (1.86)

3.7.4 Incremental Learning Results

Given a data set, we randomly select 20% of the data as the held-out test and the remaining 80% as training data used in incremental learning. Among the training data, 20% of them are randomly selected as the initial training batch, and the rest are divided into eight batches of equal size (10% each). One round of incremental learning is carried out at every newly arrived batch. We experiment with the two schemes of HLM based incremental learning, as introduced in Section 3.6. As a comparison with a baseline, the performance of the trivial batch learning method which retrains all the data that have arrived so far are also reported. The Vertex Substitution Heuristic (VSH) method is used to perform clustering for all the methods. Specifically, in HLM incremental learning scheme (II), VSH performs the weighted clustering by taking weighted distance matrix defined in Section 3.6.2.

For both of the HLM incremental learning schemes, the number of components (aka clusters) in each class in the initial training batch is set to 4. Because the size of every new batch is considerable in comparison with the current data, in HLM incremental learning scheme (I), we set the number of components in each class for every new batch to be the same as that for the current batch, that is, 4. As noted

Table 3.3: Classification error rates of distance-based classifiers (II)

Classifier	Amazon-47	Aural Sonar	Caltech-101	Color-signature
affinity k-NN	15.00 (4.77)	15.00 (6.12)	39.20 (0.86)	32.21 (3.13)
KRI k-NN (clip)	17.68 (4.75)	14.00 (6.82)	30.13 (0.42)	31.13 (3.00)
KRR k-NN (pinv)	16.10 (4.90)	15.25 (6.22)	29.90 (0.44)	31.79 (3.36)
SVM-KNN (clip)	17.56 (4.60)	13.75 (7.40)	36.82 (0.60)	31.25 (3.45)
SVM-similarities as kernel (clip)	81.34 (4.77)	13.00 (5.34)	33.49 (0.78)	34.96 (3.91)
SVM-similarities as features (linear)	76.10 (6.92)	14.25 (6.94)	38.18 (0.78)	36.71 (3.59)
P-SVM	70.12 (8.82)	14.25 (5.97)	34.23 (0.95)	33.54 (4.06)
HLM (gknn)	NA	14.50 (5.22)	52.01 (1.13)	39.79 (4.82)
HLM (agg)	NA	13.75 (6.30)	48.03 (1.29)	38.42 (4.04)
Classifier	Face Rec	Imagery	Mirex	Patrol
affinity k-NN	4.23 (1.48)	35.98 (3.63)	61.15 (1.90)	11.67 (4.08)
KRI k-NN (clip)	4.15 (1.32)	35.43 (3.32)	61.20 (2.03)	11.56 (4.54)
KRR k-NN (pinv)	4.31 (1.86)	35.91 (3.54)	61.18 (1.96)	12.81 (4.62)
SVM-KNN (clip)	4.23 (1.25)	36.13 (3.60)	61.25 (1.95)	11.98 (4.36)
SVM-similarities as kernel (clip)	4.18 (1.25)	44.23 (2.87)	57.83 (2.05)	38.75 (4.81)
SVM-similarities as features (linear)	4.29 (1.36)	48.86 (2.59)	55.54 (2.52)	42.19 (5.85)
P-SVM	4.05 (1.44)	40.93 (2.33)	63.81 (2.70)	40.42 (5.94)
HLM (gknn)	NA	39.79 (4.82)	81.10 (1.66)	NA
HLM (agg)	NA	46.27 (2.98)	61.54 (1.74)	NA
Classifier	Protein	Photo Composition	Sonar	Voting
affinity k-NN	30.81 (6.61)	25.33 (7.56)	20.36 (6.00)	5.86 (1.78)
KRI k-NN (clip)	30.35 (9.71)	22.83 (8.18)	20.00 (4.90)	5.29 (1.80)
KRR k-NN (pinv)	9.53 (5.04)	22.67 (8.79)	20.71 (5.22)	5.52 (1.69)
SVM-KNN (clip)	11.86 (5.50)	21.50 (9.57)	20.00 (5.29)	5.23 (2.25)
SVM-similarities as kernel (clip)	5.35 (4.60)	19.33 (7.64)	19.29 (6.25)	4.89 (2.05)
SVM-similarities as features (linear)	3.02 (2.76)	19.33 (7.42)	20.60 (5.60)	5.40 (2.03)
P-SVM	1.86 (1.89)	NA	19.29 (4.64)	5.34 (1.72)
HLM (gknn)	14.53 (12.59)	23.67 (8.02)	31.43 (5.81)	6.95 (2.67)
HLM (agg)	15.00 (13.00)	24.00 (8.86)	24.88 (6.18)	5.17 (2.37)

in Section 3.6, in HLM incremental learning scheme (II), to prevent old cluster centroids from staying as the only centroids, we gradually increase the number of components in each class by 2 at every learning round. To compare on a common ground this scheme and the baseline retraining scheme, we apply the latter scheme using the same number of components in each class at every learning round.

When the available training data increase, the corresponding classification error rates achieved by each method are shown in Figure 3.1. We denote the two HLM based incremental learning methods by HLM-Incremental (I) and (II). HLM-Retrain is the baseline scheme which retrains all the available data at every round.

Table 3.4: Running time of distance-based classifiers

Classifier	Amazon-47	Color-signature	Imagery	Mirex
k-NN	50 ms	50 ms	400 ms	950 ms
Local SDA	6.6 s	5 min 25.0 s	9 min 55.9 s	26 min 17.0 s
SVM-similarities as features (rbf)	10.5 s	2 min 21.3 s	20 min 4.8 s	220 min 16.8 s
HLM (vsh)	NA	40.2 s	4 min 47.8 s	11 min 1.3 s
HLM (kernel)	30 ms	280 ms	1.7 s	4.4 s

In HLM-Incremental (II), the threshold v for down sampling in any cluster is 4. Similarly as in the previous experiment setup and computing environment, clusters containing less than three data points are not used in the parameter estimation of HLM and all the experiments are conducted on a 2.66 GHz Intel CPU. Table 3.5 shows the total running time across all the learning rounds for each of the incremental learning methods. HLM-Incremental (I) has the shortest running time on all the data sets, HLM-Incremental (II) being the second fastest, while HLM-Retrain is the slowest. The proportional reduction in computation between HLM-Incremental (I)/(II) and HLM-Retrain is most prominent with the largest data set, Mirex.

Based on Figure 3.1, for the Mirex data and the Color-signature data before the last two learning rounds, when the available training data increase, the classification error rates have a clear decreasing trend. For these two data sets, HLM-Retrain stays as the winner at most of the learning rounds. However, it is interesting to note that for the Voting data, HLM-Incremental (I), albeit being the fastest, is the winner among the three across all the rounds; and HLM-Incremental (II) outperforms HLM-Retrain at several rounds. For Mirex and Color-signature, HLM-Incremental (II), the second fastest scheme, performs slightly worse than HLM-Retrain at most of the learning rounds; and the fastest scheme HLM-Incremental (I) performs worst at almost all the learning rounds. This observation indicates a trade-off between computational intensity and performance. For all the data sets except Color-signature, less than 5% difference is observed between the classification error rates obtained by HLM-Incremental (II) and HLM-Retrain at the last learning round. HLM-Incremental (II) even performs slightly better than HLM-Retrain on the Mirex and Voting data at the last learning round. For the Imagery

and Voting data, the error rate curves under all the methods fluctuate, and at the last learning round, the error rates are only slightly better or even worse than where they start off at the first round. The fluctuations of the error rate curves are also observed at the last two learning rounds for Color-signature. We believe that such counter intuitive outcomes are due to the inherent randomness of the data.

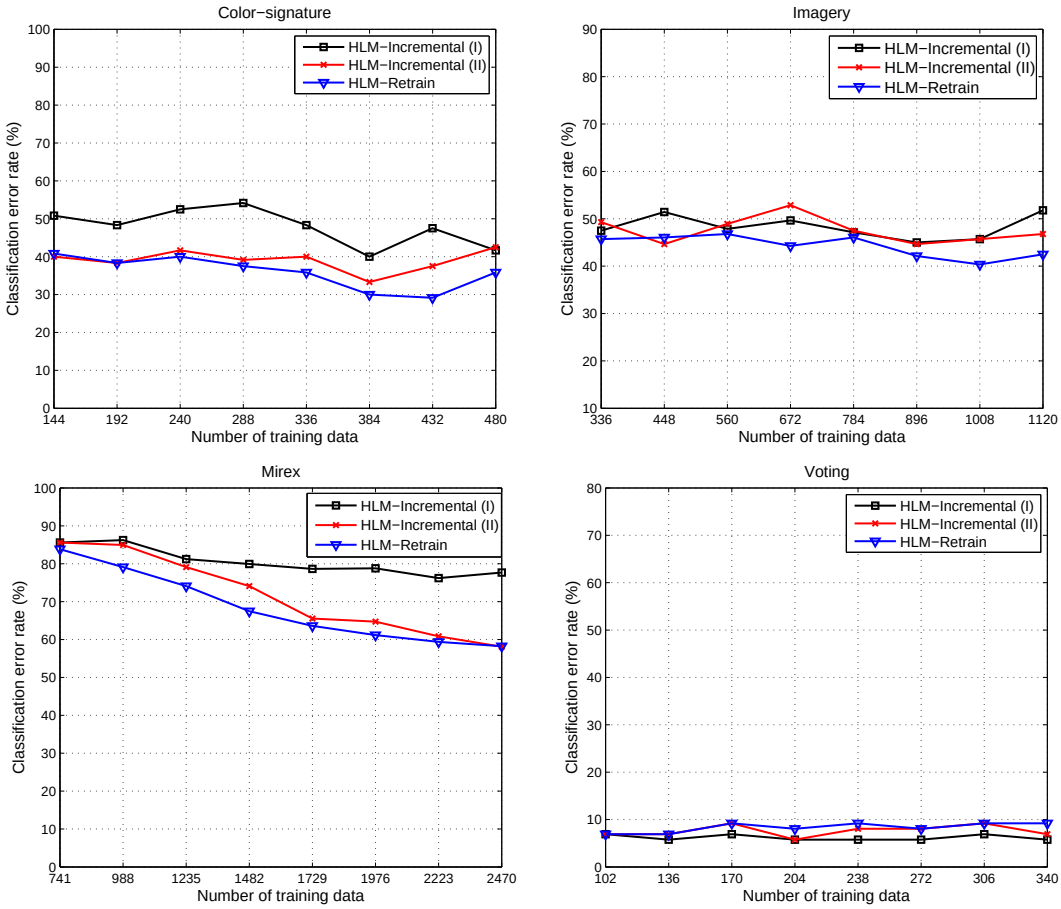


Figure 3.1: The classification error rates of incremental and batch learning methods at each learning round as the number of available training data increases.

Table 3.5: Running time of HLM based incremental learning methods (seconds)

Classifier	Color-signature	Imagery	Mirex	Voting
HLM-Incremental (I)	0.22	0.40	0.84	0.14
HLM-Incremental (II)	1.06	2.48	2.87	0.59
HLM-Retrain	1.72	12.54	22.89	2.04

3.8 Summary

A distance-based mixture modeling approach based on hypothetical local mapping (HLM) is proposed. Because only pairwise distances are needed, HLM is particularly useful for data that cannot be easily described by a mathematical entity. The application of the proposed mixture model to classification is explored. We have compared this approach with several other state-of-the-art distance-based classification methods on various datasets. Experimental results show that HLM based algorithms perform competitively at low computational cost during both training and testing. Because a mixture model is estimated for each class separately, scalability is achieved for a large number of classes. HLM adapts readily to learning a classifier in an incremental fashion. None of the local SDA, SVM, and k-NN based classifiers can be easily modified for incremental learning. We have proposed two incremental learning schemes for HLM and found that they perform closely to the baseline of retraining over all the available data, but at a much faster speed.

Chapter 4

Gaussian Mixture Models with Component Means Constrained in Pre-selected Subspaces

4.1 Introduction

The Gaussian mixture model (GMM) is a popular and effective tool for clustering and classification. When applied to clustering, usually each cluster is modeled by a Gaussian distribution. Because the cluster labels are unknown, we face the issue of estimating a GMM. A thorough treatment of clustering by GMM is referred to (McLachlan and Peel, 2000a). Hastie and Tibshirani (1996) proposed the mixture discriminant analysis (MDA) for classification, which assumes a GMM for each class. Fraley and Raftery (2002) examined the roles of GMM for clustering, classification, and density estimation.

As a probability density, GMM enjoys great flexibility comparing with parametric distributions. Although GMM can approximate any smooth density by increasing the number of components R, the number of parameters in the model grows quickly with R, especially for high dimensional data. The regularization of GMM has been a major research topic on mixture models. Early efforts focused on controlling the complexity of the covariance matrices, partly driven by the frequent occurrences of singular matrices in estimation (Fraley and Raftery, 2002). More

recently, it is noted that for data with very high dimensions, a mixture model with parsimonious covariance structures, for instance, common diagonal matrices, may still have high complexity due to the component means alone. Methods to regularize the component means have been proposed from quite different perspectives. Li and Zha (2006) developed the so-called two-way mixture of Poisson distributions in which the variables are grouped and the means of the variables in the same group within any component are assumed identical. The grouping of the variables reduces the number of parameters in the component means dramatically. In the same spirit, Qiao and Li (2010) developed the two-way mixture of Gaussians. Pan and Shen (2007) explored the penalized likelihood method with L_1 norm penalty on the component means. The method aims at shrinking the component means of some variables to a common value. Variable selection for clustering is achieved because the variables with common means across all the components are non-informative for cluster labels. Wang and Zhu (2008) proposed the L_∞ norm as a penalty instead.

In this chapter, we propose another approach to regularizing the component means in GMM, which is more along the line of reduced rank MDA (Hastie and Tibshirani, 1996) but with profound differences. We search for a linear subspace in which the component means reside and estimate a GMM under such a constraint. The constrained GMM has a dimension reduction property. It is proved that with the subspace restriction on the component means and under common covariance matrices, only a linear projection of the data with the same dimension as the subspace matters for classification and clustering. The method is especially useful for visualization when we want to view data in a low dimensional space which best preserves the classification and clustering characteristics.

The idea of restricting component means to a linear subspace was first explored in the linear discriminant analysis (LDA). Fisher (1936) proposed to find a subspace of rank $r < K$, where K is the number of classes, so that the projected class means are spread apart maximally, The coordinates of the optimal subspace are derived by successively maximizing the between-class variance relative to the within-class variance, known as *canonical* or *discriminant* variables. Although LDA does not involve the estimation of a mixture model, the marginal distribution of the observation without the class label is a mixture distribution. The idea of reduced rank

LDA was used by Hastie and Tibshirani (1996) for GMM. It was proved in Hastie and Tibshirani (1996) that reduced rank LDA can be viewed as a Gaussian maximum likelihood solution with the restriction that the means of Gaussians lie in a L-dimension subspace, i.e., $\text{rank}\{\mu_k\}_1^K = L < \max(K-1, p)$, where μ_k's are the means of Gaussians and p is the dimension of the data. Hastie and Tibshirani (1996) extended this concept and proposed a reduced rank version of the mixture discriminant analysis (MDA), which performed a reduced rank weighted LDA in each iteration of the EM algorithm.

Another related line of research is regularizing the component means of a GMM in a latent factor space, i.e., the use of factor analysis in GMM. It was originally proposed by Ghahramani and Hinton (1997) to perform concurrent clustering and dimension reduction using mixture of factor analyzers; see also McLachlan and Peel (2000b) and McLachlan et al. (2003). Factor analysis was later used to regularize the component means of a GMM in each state of the Hidden Markov Model (HMM) for speaker verification (Kenny et al., 2008; Povey et al., 2011). In those models, the total number of parameters is significantly reduced due to the regularization, which effectively prevents over fitting. Usually the EM type of algorithm is applied to estimate the parameters and find the latent subspace.

The role of the subspace constraining the means differs intrinsically between our approach, the reduced rank MDA and factor analysis based mixture models, resulting in mathematical solutions of quite different nature. Within each iteration of the EM algorithm for estimating a GMM, the reduced rank MDA finds the subspace with a given dimension that yields the maximum likelihood under the current partition of the data into the mixture components. The subspace depends on the component-based clustering of data in each iteration. Similarly, the subspaces in factor analysis based mixture models are found through the iterations of the EM algorithm, as part of the model estimation. However, in our method, we treat the seek of the subspace and the estimation of the model separately. The subspace is fixed throughout the estimation of the GMM. Mathematically speaking, we try to solve the maximum likelihood estimation of GMM under the condition that the component means lie in a given subspace.

Our formulation of the model estimation problem allows us to exploit multiple and better choices of density estimate when we seek the constraining subspace. For

instance, if we want to visualize the data in a plane while the component means are not truly constrained to a plane, fitting a GMM with means constrained to a plane may lead to poor density estimation. As a result, the plane sought during the estimation will be problematic. It is thus sensible to find the plane based on a density estimate without the constraint. Afterward, we can fit a GMM under the constraint purely for the purpose of visualization. Moreover, the subspace may be specified based on prior knowledge. For instance, in multi-dimensional data visualization, we may already know that the component (or cluster) means of data lie in a subspace spanned by several dimensions of the data. Therefore, the subspace is required to be fixed.

We propose two approaches to finding the unknown subspace. The first approach is the so-called *modal PCA (MPCA)*. We prove that the modes (local maxima) lie in the same constrained subspace as the component means. We use the *modal EM (MEM)* algorithm (Li et al., 2007) to find the modes. By exploiting the modes, we are no longer restricted to the GMM as a tool for density estimation. Instead, we use the kernel density estimate which avoids sensitivity to initialization. There is an issue of choosing the bandwidth, which is easier than usual in our framework by the following strategy. We take a sequence of subspaces based on density estimates resulting from different kernel bandwidths. We then estimate GMMs under the constraint of each subspace and finally choose a model yielding the maximum likelihood. Note that, each GMM is a full model for the original data, although the component means are constrained in a different subspace. We therefore can compare the estimated likelihood under each model. This framework in fact extends beyond kernel density estimation. As discussed in (Li et al., 2007), modes can be found using modal EM for any density in the form of a mixture distribution. The second approach is an extension of MPCA which exploits class means or a union set of modes and class means. It is easy to see that the class means also reside in the same constrained subspace as the component means. Comparing with modes, class means do not depend on the kernel bandwidth and are more robust to estimate.

Experiments on the classification of several real data sets with moderate to high dimensions show that reduced rank MDA does not always have good performance. We do not intend to claim that our proposed method is necessarily better than

reduced rank MDA. However, when the constraining subspace of the component means is of a very low dimension, we find that the proposed method with the simple technique of finding the subspace based on class means often outperforms the reduced rank MDA, which solves a discriminant subspace using a much more sophisticated approach. In addition, we compare our methods with standard MDA on the data projected to the subspace containing the component means. For data with moderately high dimensions, our proposed method works better. Besides classification, our method easily applies to clustering.

The rest of the chapter is organized as follows. In Section 4.2, we review some background and notation. We present a Gaussian mixture model with subspace constrained component means, the MPCA algorithm and its extension for finding the subspace in Section 4.3. We also present several properties of the constrained subspace, with detailed proofs in the appendix. In Section 4.4, we describe the estimation algorithm for the proposed model. Experimental results are provided in Section 4.5. Finally, we conclude and discuss future work in Section 4.6.

4.2 Preliminaries and Notation

Let $\boldsymbol{X} = (X_1, X_2, ..., X_p)^t$, where p is the dimension of the data. A sample of $\boldsymbol{X}$ is denoted by $\boldsymbol{x} = (x_1, x_2, ..., x_p)^t$. We present the notations for a general Gaussian mixture model before introducing the mixture model with component means constrained to a given subspace. Gaussian mixture model can be applied to both classification and clustering. Let the class label of $\boldsymbol{X}$ be $Y \in \mathcal{K} = \{1, 2, ..., K\}$. For classification purpose, the joint distribution of $\boldsymbol{X}$ and Y under a Gaussian mixture is

$$f(\boldsymbol{X} = \boldsymbol{x}, Y = k) = a_k f_k(\boldsymbol{x}) = a_k \sum_{r=1}^{R_k} \pi_{kr} \phi(\boldsymbol{x} | \boldsymbol{\mu}_{kr}, \boldsymbol{\Sigma}) \ , \tag{4.1}$$

where a_k is the prior probability of class k, satisfying $0 \leq a_k \leq 1$ and $\sum_{k=1}^{K} a_k = 1$, and $f_k(\boldsymbol{x})$ is the within-class density for $\boldsymbol{X}$. R_k is the number of mixture components used to model class k, and the total number of mixture components for all the classes is $R = \sum_{k=1}^{K} R_k$. Let π_{kr} be the mixing proportions for the rth component in class k, $0 \leq \pi_{kr} \leq 1$, $\sum_{r=1}^{R_k} \pi_{kr} = 1$. $\phi(\cdot)$ denotes the pdf

of a Gaussian distribution: $\boldsymbol{\mu}_{kr}$ is the mean vector for component r in class k and $\boldsymbol{\Sigma}$ is the common covariance matrix shared across all the components in all the classes. To classify a sample $\boldsymbol{X} = \boldsymbol{x}$, the Bayes classification rule is used: $\widehat{y} = \text{argmax}_k f(Y = k|\boldsymbol{X} = \boldsymbol{x}) = \text{argmax}_k f(\boldsymbol{X} = \boldsymbol{x}, Y = k)$.

In the context of clustering, the Gaussian mixture model is now simplified as

$$f(\boldsymbol{X} = \boldsymbol{x}) = \sum_{r=1}^{R} \pi_r \phi(\boldsymbol{x}|\boldsymbol{\mu}_r, \boldsymbol{\Sigma}) \ , \tag{4.2}$$

where R is the total number of mixture components and π_r is the mixing proportions for the rth component. $\boldsymbol{\mu}_r$ and $\boldsymbol{\Sigma}$ denote the rth component mean and the common covariance matrix for all the components. The clustering procedure involves first fitting the above mixture model and then computing the posterior probability of each mixture component given a sample point. The component with the highest posterior probability is chosen for that sample point, and all the points belonging to the same component form one cluster.

In this work, we assume that the Gaussian component means reside in a given linear subspace and estimate a GMM with subspace constrained means. A new algorithm, namely the *modal PCA (MPCA)*, is proposed to find the constrained subspace. The motivations of using modes to find subspace are outlined in Section 4.3.1. Before we present MPCA, we will first introduce the modal EM algorithm (Li et al., 2007) which solves the local maxima, that is, modes, of a mixture density.

Modal EM: Given a mixture density $f(\boldsymbol{X} = \boldsymbol{x}) = \sum_{r=1}^{R} \pi_r f_r(\boldsymbol{x})$, as in model (4.2), starting from any initial data point $\boldsymbol{x}^{(0)}$, the modal EM algorithm finds a mode of the density by alternating the following two steps until a stopping criterion is met. Start with $t = 0$.

1. Let $p_r = \frac{\pi_r f_r(\boldsymbol{x}^{(t)})}{f(\boldsymbol{x}^{(t)})}$, $r = 1, ..., R$.
2. Update $\boldsymbol{x}^{(t+1)} = \text{argmax}_{\boldsymbol{x}} \sum_{r=1}^{R} p_r \log f_r(\boldsymbol{x})$.

The above two steps are similar to the expectation and the maximization steps in EM (Dempster et al., 1977). The first step is the "expectation" step where the posterior probability of each mixture component r, $1 \leq r \leq R$, at the current data point $\boldsymbol{x}^{(t)}$ is computed. The second step is the "maximization" step.

$\sum_{r=1}^{R} p_r \log f_r(\boldsymbol{x})$ has a unique maximum, if the $f_r(\boldsymbol{x})$'s are normal densities. In the special case of a mixture of Gaussians with common covariance matrix, that is, $f_r(\boldsymbol{x}) = \phi(\boldsymbol{x} \mid \boldsymbol{\mu}_r, \boldsymbol{\Sigma})$, we simply have $\boldsymbol{x}^{(t+1)} = \sum_{r=1}^{R} p_r \boldsymbol{\mu}_r$. In modal EM, the probability density function of the data is estimated nonparametrically using Gaussian kernels, which are in the form of a Gaussian mixture distribution:

$$f(\boldsymbol{X} = \boldsymbol{x}) = \sum_{i=1}^{n} \frac{1}{n} \phi(\boldsymbol{x} \mid \boldsymbol{x}_i, \boldsymbol{\Sigma}) \,,$$

where the Gaussian density function is

$$\phi(\boldsymbol{x} \mid \boldsymbol{x}_i, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2}} \exp(-\frac{1}{2} (\boldsymbol{x} - \boldsymbol{x}_i)^t \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{x}_i)) \,.$$

We use a spherical covariance matrix $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I}_p$. The standard deviation σ is also referred to as the *bandwidth* of the Gaussian kernel. When the bandwidth of Gaussian kernels increases, the density estimate becomes smoother, and more data points tend to ascend to the same mode. Different numbers of modes can thus be found by gradually increasing the bandwidth of Gaussian kernels. The data points are grouped into one cluster if they climb to the same mode. We call the mode as the cluster representative.

In (Li et al., 2007), a hierarchical clustering approach, namely, *Hierarchical Mode Association Clustering (HMAC)*, is proposed based on mode association and kernel bandwidth growth. Given a sequence of bandwidths $\sigma_1 < \sigma_2 < \cdots < \sigma_\eta$, HMAC starts with every point $\boldsymbol{x_i}$ being a cluster by itself, which corresponds to the extreme case that σ_1 approaches 0. At any bandwidth $\sigma_l (l > 1)$, the modes, that is, cluster representatives, obtained from the preceding bandwidth are input into the modal EM algorithm. The modes identified then form a new set of cluster representatives. This procedure is repeated across all σ_l's. For details of HMAC, we refer interested readers to (Li et al., 2007). We therefore obtain modes at different levels of bandwidth by HMAC. The clustering performed by HMAC is only for the purpose of finding modes across different bandwidths and should not be confused with the clustering or classification based on the Gaussian mixture model we propose here.

4.3 GMM with Subspace Constrained Means

The Gaussian mixture model with subspace constrained means is presented in this section. For brevity, we focus on the constrained mixture model in a classification set-up, since clustering can be treated as a "one-class" modeling and is likewise solved.

We propose to model the within-class density by a Gaussian mixture with component means constrained to a pre-selected subspace:

$$f_k(\boldsymbol{x}) = \sum_{r=1}^{R_k} \pi_{kr} \phi(\boldsymbol{x}|\boldsymbol{\mu}_{kr}, \boldsymbol{\Sigma}) \tag{4.3}$$

subject to

$$\boldsymbol{v}_j^t \cdot \boldsymbol{\mu}_{k1} = \boldsymbol{v}_j^t \cdot \boldsymbol{\mu}_{k2} = \cdots = \boldsymbol{v}_j^t \cdot \boldsymbol{\mu}_{kR_k} = c_j \ , \tag{4.4}$$

where $\boldsymbol{v}_j$'s are linearly independent vectors, $j = 1, ..., q$, $q < p$, and c_j is a constant, invariant to different classes. Without loss of generality, we can assume $\{\boldsymbol{v}_1, ..., \boldsymbol{v}_q\}$ span an orthonormal basis. Augment it to full rank by $\{\boldsymbol{v}_{q+1}, ..., \boldsymbol{v}_p\}$. Suppose $\boldsymbol{\nu} = \{\boldsymbol{v}_{q+1}, ..., \boldsymbol{v}_p\}$, $\boldsymbol{\nu}^\perp = \{\boldsymbol{v}_1, ..., \boldsymbol{v}_q\}$, and $\boldsymbol{c} = (c_1, c_2, ..., c_q)^t$. Denote the projection of a vector $\boldsymbol{\mu}$ or a matrix U onto an orthonormal basis S by $\mathbf{Proj}_S^{\boldsymbol{\mu}}$ or $\mathbf{Proj}_S^U$. We have $\mathbf{Proj}_{\boldsymbol{\nu}^\perp}^{\boldsymbol{\mu}_{kr}} = \boldsymbol{c}$ over all the k and r. That is, the projections of all the component means $\boldsymbol{\mu}_{kr}$'s onto the orthonormal basis $\boldsymbol{\nu}^\perp$ coincide at $\boldsymbol{c}$. We refer to $\boldsymbol{\nu}$ as the *constrained subspace*[1] where $\boldsymbol{\mu}_{kr}$'s reside (or more strictly, $\boldsymbol{\mu}_{kr}$'s reside in the subspace up to a translation), and $\boldsymbol{\nu}^\perp$ as the corresponding *null subspace*. Suppose the dimension of the constrained subspace $\boldsymbol{\nu}$ is d, then $d = p - q$. With the constraint (4.4) and the assumption of a common covariance matrix across all the components in all the classes, essentially, we assume that the data within each component have identical distributions in the null space $\boldsymbol{\nu}^\perp$. In the following section, we will explain how to find an appropriate constrained subspace $\boldsymbol{\nu}$.

[1]We abuse the notation $\boldsymbol{\nu}$ slightly here. The subspace is actually spanned by $\boldsymbol{\nu}$. To simplify notations, the same abuse also appears in other similar scenarios.

4.3.1 Modal PCA

We introduce in this section the modal PCA (MPCA) algorithm that finds a constrained subspace for the component means of a Gaussian mixture and the properties of the found subspace. We prove in Appendix B.1 the following theorem.

Theorem 4.3.1. *For a Gaussian mixture model with component means constrained in a subspace* $\boldsymbol{\nu} = \{\boldsymbol{v}_{q+1}, ..., \boldsymbol{v}_p\}$*,* $q < p$*, and a common covariance matrix across all the components, the modes of the mixture density are also constrained in the same subspace* $\boldsymbol{\nu}$*.*

According to Theorem 4.3.1, the modes and component means of Gaussian mixtures reside in the same constrained subspace. We use the aforementioned MEM algorithm introduced in Section 4.2 to find the modes of the density. To avoid sensitivity to initialization and the number of components, we use the Gaussian kernel density estimate instead of a finite mixture model for the density. It is well known that mixture distributions with drastically different parameters may yield similar densities. We are thus motivated to exploit modes which are geometric characteristics of the densities.

Let us denote the set of modes found by MEM under the kernel bandwidth σ by $\mathcal{G} = \{\mathcal{M}_{\sigma,1}, \mathcal{M}_{\sigma,2}, ..., \mathcal{M}_{\sigma,|\mathcal{G}|}\}$. A weighted principal component analysis is proposed to find the constrained subspace. A weight $w_{\sigma,r}$ is assigned to the rth mode, which is the proportion of sample points in the entire data ascending to that mode. We therefore have a weighted covariance matrix of all the modes in $\mathcal{G}$:

$$\Sigma_{\mathcal{G}} = \sum_{r=1}^{|\mathcal{G}|} w_{\sigma,r}(\mathcal{M}_{\sigma,r} - \mu_{\mathcal{G}})^T(\mathcal{M}_{\sigma,r} - \mu_{\mathcal{G}}) \ ,$$

where $\mu_{\mathcal{G}} = \sum_{r=1}^{|\mathcal{G}|} w_{\sigma,r}\mathcal{M}_{\sigma,r}$. The principal components are then obtained by performing an eigenvalue decomposition on $\Sigma_{\mathcal{G}}$. Recall the dimension of the constrained subspace $\boldsymbol{\nu}$ is d. Since the leading principal components capture the most variation in the data, we use the first d most significant principal components to span the constrained subspace $\boldsymbol{\nu}$, and the remaining principal components to span the corresponding null space $\boldsymbol{\nu}^{\perp}$.

Given a sequence of bandwidths $\sigma_1 < \sigma_2 < \cdots < \sigma_\eta$, the modes at different levels of bandwidth can be obtained using the HMAC algorithm introduced in

Section 4.2. At each level, we apply the weighted PCA to the modes, and obtain a new constrained subspace by their first d most significant principal components. In practice, if the number of modes found at a particular level of bandwidth is smaller than 3, we will skip the modes at that level. For the extreme case, when $\sigma = 0$, the subspace is actually spanned by the principal components of the original data points. We therefore obtain a collection of subspaces, $\boldsymbol{\nu}_1, ..., \boldsymbol{\nu}_\eta$, resulting from a sequence of bandwidths through HMAC.

4.3.2 Extension of Modal PCA

In this section, we propose another approach to generating the constrained subspace, which is an extension of MPCA. Suppose the mean of class k is $\mathcal{M}'_k$, we have $\mathcal{M}'_k = \sum_{r=1}^{R_k} \pi_{kr} \boldsymbol{\mu}_{kr}$, where $\boldsymbol{\mu}_{kr}$ is the rth component in class k. It is easy to see that the class means lie in the same subspace as the Gaussian mixture component means. From Theorem 4.3.1, we know that in Gaussian mixtures, the modes and component means also reside in the same constrained subspace. So the class means, modes and component means all lie in the same constrained subspace. Comparing with the modes, class means are more robust to estimate. It is thus natural to incorporate class means to find the subspace. In the new approach, if the dimension d of the constrained subspace is smaller than K, the subspace is spanned by applying weighted PCA only to class means. Otherwise, it is spanned by applying weighted PCA to a union set of modes and class means.

Similar to modal PCA, we first assign a weight a_k to the kth class mean $\mathcal{M}'_k$, which is the proportion of the number of sample points in class k over the entire data, i.e., the prior probability of class k. Suppose the set of class means is $\mathcal{J} = \{\mathcal{M}'_1, \mathcal{M}'_2, \cdots, \mathcal{M}'_K\}$. If $d < K$, we have a weighted covariance matrix of all the class means:

$$\Sigma_{\mathcal{J}} = \sum_{r=1}^{K} a_k (\mathcal{M}'_r - \mu_{\mathcal{J}})^T (\mathcal{M}'_r - \mu_{\mathcal{J}}) \ ,$$

where $\mu_{\mathcal{J}} = \sum_{r=1}^{K} a_k \mathcal{M}'_k$. An eigenvalue decomposition on $\Sigma_{\mathcal{J}}$ is then performed to obtain all the principal components. Similar to MPCA, the constrained subspace is spanned by the first d most significant principal components. If $d \geq K$, we will put together all the class means and modes and assign different weights to them. Suppose γ is a value between 0 and 100, we allocate a total of $\gamma\%$ of weight to the

class means, and the remaining $(100-\gamma)\%$ weights allocated proportionally to the modes. That is, the weights assigned to the class mean $\mathcal{M}'_k$ and the mode $\mathcal{M}_{\sigma,r}$ are $\gamma a_k\%$ and $(100-\gamma)w_{\sigma,r}\%$, respectively. Then the weighted covariance matrix of the union set of class means and modes becomes

$$\begin{aligned}\Sigma_{\mathcal{G}\cup\mathcal{J}} &= \sum_{r=1}^{K}(\gamma a_k\%)(\mathcal{M}'_r-\mu_{\mathcal{J}})^T(\mathcal{M}'_r-\mu_{\mathcal{J}}) \\ &\quad + \sum_{r=1}^{|\mathcal{G}|}((100-\gamma)w_{\sigma,r}\%)(\mathcal{M}_{\sigma,r}-\mu_{\mathcal{G}})^T(\mathcal{M}_{\sigma,r}-\mu_{\mathcal{G}})\ .\end{aligned}$$

Different weights can be allocated to the class means and the modes. For instance, if we want the class means to play a more important role in spanning subspaces, we can set $\gamma > 50$. Again, an eigenvalue decomposition is performed on $\Sigma_{\mathcal{G}\cup\mathcal{J}}$ to obtain all the principal components and the first d most significant principal components span the constrained subspace. To differentiate this method from MPCA, we denote it by MPCA-MEAN.

4.3.3 Dimension Reduction

The mixture model with component means under constraint (4.4) implies a dimension reduction property for the classification purpose, formally stated below.

Theorem 4.3.2. *For a Gaussian mixture model with a common covariance matrix* $\boldsymbol{\Sigma}$*, suppose all the component mean* $\boldsymbol{\mu}_{kr}$*'s are constrained in a subspace spanned by* $\boldsymbol{\nu} = \{\boldsymbol{v}_{q+1}, ..., \boldsymbol{v}_p\}$*,* $q < p$*, up to a translation, only a linear projection of the data* $\boldsymbol{x}$ *onto a subspace spanned by* $\{\boldsymbol{\Sigma}^{-1}\boldsymbol{v}_j | j = q+1, ..., p\}$ *(the same dimension as* $\boldsymbol{\nu}$*) is informative for classification.*

In Appendix B.2, we provide the detailed proof for Theorem 4.3.2. If the common covariance matrix $\boldsymbol{\Sigma}$ is an identity matrix (or a scalar matrix), the class label Y only depends on the projection of $\boldsymbol{x}$ onto the constrained subspace $\boldsymbol{\nu}$. However, in general, $\boldsymbol{\Sigma}$ is non-identity. Hence the spanning vectors, $\boldsymbol{\Sigma}^{-1}\boldsymbol{v}_j$, $j = q+1, ..., p$, for the subspace informative for classification are not orthogonal in general as well. In Appendix B.2, we use the column vectors of $orth(\{\boldsymbol{\Sigma}^{-1}\boldsymbol{v}_j | j = q+1, ..., p\})$ to span this subspace. To differentiate it from the constrained subspace in which the component means lie, we call it as *discriminant subspace*. The dimension

of the discriminant subspace is referred to as *discriminant dimension*, which is the dimension actually needed for classification. The discriminant subspace is of the same dimension as the constrained subspace. When the discriminant dimension is small, significant dimension reduction is achieved. Our method can thus be used as a data reduction tool for visualization when we want to view the classification of data in a two or three dimensional space.

Although in Appendix B.2 we prove Theorem 4.3.2 in the context of classification, the proof can be easily modified to show that the dimension reduction property applies to clustering as well. That is, we only need the data projected onto a subspace with the same dimension as the constrained subspace $\boldsymbol{\nu}$ to compute the posterior probability of the data belonging to a component (aka cluster). Similarly, we name the subspace that matters for clustering as *discriminant subspace* and its dimension as *discriminant dimension*.

4.4 Model Estimation

We will first describe in Section 4.4.1 the basic version of the estimation algorithm where the constraints on the component means are characterized by (4.4). A natural extension to the constraint in (4) is to allow the constant c_j to vary with the class labels, thus leading to constraint characterized in (4.10). The corresponding algorithm is described in Section 4.4.2.

4.4.1 The Algorithm

Let us first summarize the work flow of our proposed method:

1. Given a sequence of kernel bandwidths $\sigma_1 < \sigma_2 < \cdots < \sigma_\eta$, apply HMAC to find the modes of the density estimation at each bandwidth σ_l.
2. Apply MPCA or MPCA-MEAN to the modes or a union set of modes and class means at each kernel bandwidth and obtain a sequence of constrained subspaces.
3. Estimate the Gaussian mixture model with component means constrained in each subspace and select the model yielding the maximum likelihood.

4. Perform classification on the test data or clustering on the overall data, with the selected model from Step 3.

Remarks:

1. In our method, the seek of subspace and the estimation of the mixture model are separate. We first search for a sequence of subspaces and then estimate the model constrained in each subspace separately.
2. In Step 1, the identified modes are from the density estimation of the overall data (in clustering) or the overall training data (in classification).
3. For MPCA-MEAN, if the dimension d of the constrained subspace is smaller than K, the subspace is spanned only by class means and is therefore fixed. We do not need to choose the subspace.
4. Some prior knowledge may be exploited to yield an appropriate subspace. Then, we can estimate GMM under the constraint of the given subspace directly.

Now we will derive an EM algorithm to estimate a GMM under the constraint of a given subspace. The estimation method for classification is introduced first. A common covariance matrix $\mathbf{\Sigma}$ is assumed across all the components in all the classes. In class k, the parameters to be estimated include the class prior probability a_k, the mixture component prior probabilities π_{kr}, and the Gaussian parameters $\boldsymbol{\mu}_{kr}$, $\mathbf{\Sigma}$, $r = 1, 2, ..., R_k$. Denote the training data by $\{(\boldsymbol{x}_i, y_i) : i = 1, ..., n\}$. Let n_k be the number of data points in class k. The total number of data points n is $\sum_{k=1}^{K} n_k$. The class prior probability a_k is estimated by the empirical frequency $n_k / \sum_{k'=1}^{K} n_{k'}$. The EM algorithm comprises the following two steps:

1. *Expectation-step*: Given the current parameters, for each class k, compute the component posteriori probability for each data point $\boldsymbol{x}_i$ within class k:

$$q_{i,kr} \propto \pi_{kr} \phi(\boldsymbol{x}_i | \boldsymbol{\mu}_{kr}, \mathbf{\Sigma}) \ , \quad \text{subject to} \sum_{r=1}^{R_k} q_{i,kr} = 1 \ . \tag{4.5}$$

2. *Maximization-step*: Update $\pi_{kr}, \boldsymbol{\mu}_{kr}$, and $\mathbf{\Sigma}$, which maximize the following objective function (the i subscript indicates $\boldsymbol{x}_i$ with $y_i = k$):

$$\sum_{k=1}^{K} \sum_{r=1}^{R_k} \left(\sum_{i=1}^{n_k} q_{i,kr} \right) \log \pi_{kr} + \sum_{k=1}^{K} \sum_{r=1}^{R_k} \sum_{i=1}^{n_k} q_{i,kr} \log \phi(\boldsymbol{x}_i | \boldsymbol{\mu}_{kr}, \mathbf{\Sigma}) \tag{4.6}$$

under the constraint (4.4).

In the maximization step, the optimal π_{kr}'s are not affected by the constraint (4.4) and are solved separately from $\boldsymbol{\mu}_{kr}$'s and $\boldsymbol{\Sigma}$:

$$\pi_{kr} \propto \sum_{i=1}^{n_k} q_{i,kr} \ , \quad \sum_{r=1}^{R_k} \pi_{kr} = 1 \ . \tag{4.7}$$

Since there are no analytic solutions to $\boldsymbol{\mu}_{kr}$'s and $\boldsymbol{\Sigma}$ in the above constrained optimization, we adopt the generalized EM (GEM) algorithm. Specifically, we use a conditional maximization approach. In every maximization step of GEM, we first fix $\boldsymbol{\Sigma}$, and then update the $\boldsymbol{\mu}_{kr}$'s. Then we update $\boldsymbol{\Sigma}$ conditioned on the $\boldsymbol{\mu}_{kr}$'s held fixed. This iteration will be repeated multiple times.

Given $\boldsymbol{\Sigma}$, solving $\boldsymbol{\mu}_{kr}$ is non-trivial. The key steps are summarized here. For detailed derivation, we refer interested readers to Appendix B.3. In constraint (4.4), we have $\boldsymbol{v}_j^t \cdot \boldsymbol{\mu}_{kr} = c_j$, i.e., identical across all the k and r for $j = 1, ..., q$. It is easy to see that $\boldsymbol{c} = (c_1, ..., c_q)^t$ is equal to the projection of the mean of the overall data onto the null space $\boldsymbol{\nu}^{\perp}$. However, in practice, we do not need the value of $\boldsymbol{c}$ in the parameter estimation. Before we give the equation to solve $\boldsymbol{\mu}_{kr}$, let us define some notations first. Assume $\boldsymbol{\Sigma}$ is non-singular and hence positive definite, we can write $\boldsymbol{\Sigma} = (\boldsymbol{\Sigma}^{\frac{1}{2}})^t(\boldsymbol{\Sigma}^{\frac{1}{2}})$, where $\boldsymbol{\Sigma}^{\frac{1}{2}}$ is of full rank. If the eigen decomposition of $\boldsymbol{\Sigma}$ is $\boldsymbol{\Sigma} = V_{\boldsymbol{\Sigma}} D_{\boldsymbol{\Sigma}} V_{\boldsymbol{\Sigma}}^t$, then $\boldsymbol{\Sigma}^{\frac{1}{2}} = D_{\boldsymbol{\Sigma}}^{\frac{1}{2}} V_{\boldsymbol{\Sigma}}^t$. Let $\boldsymbol{V}_{null}$ be a $p \times q$ orthonormal matrix $(\boldsymbol{v}_1, ..., \boldsymbol{v}_q)$, the column vectors of which span the null space $\boldsymbol{\nu}^{\perp}$. Suppose $\boldsymbol{B} = \boldsymbol{\Sigma}^{\frac{1}{2}} \boldsymbol{V}_{null}$. Perform a singular value decomposition (SVD) on $\boldsymbol{B}$, i.e., $\boldsymbol{B} = \boldsymbol{U_B} \boldsymbol{D_B} \boldsymbol{V_B^t}$, where $\boldsymbol{U_B}$ is a $p \times q$ matrix, the column vectors of which form an orthonormal basis for the space spanned by the column vectors of $\boldsymbol{B}$. Let $\hat{\boldsymbol{U}}$ be a column augmented orthonormal matrix of $\boldsymbol{U_B}$. Denote $\sum_{i=1}^{n_k} q_{i,kr}$ by l_{kr}. Let $\bar{\boldsymbol{x}}_{kr} = \sum_{i=1}^{n_k} q_{i,kr} \boldsymbol{x}_i / l_{kr}$, i.e., the weighted sample mean of the component r in class k, and $\breve{\boldsymbol{x}}_{kr} = \hat{\boldsymbol{U}}^t \left(\boldsymbol{\Sigma}^{-\frac{1}{2}} \right)^t \cdot \bar{\boldsymbol{x}}_{kr}$. Define $\breve{\boldsymbol{\mu}}_{kr}^*$ by the following Eqs. (4.8) and (4.9):

1. for the first q coordinates, $j = 1, ..., q$:

$$\breve{\mu}_{kr,j}^* = \frac{\sum_{k'=1}^{K} \sum_{r'=1}^{R_{k'}} l_{k'r'} \breve{x}_{k'r',j}}{n} \ , \quad \text{identical over } r \text{ and } k \ ; \tag{4.8}$$

2. for the remaining $p-q$ coordinates, $j = q+1, ..., p$:

$$\breve{\mu}^*_{kr,j} = \breve{x}_{kr,j} \ . \tag{4.9}$$

That is, the first q constrained coordinates are optimized using component-pooled sample mean (components from all the classes) while those $p-q$ unconstrained coordinates are optimized separately within each component using the component-wise sample mean. Note that we abuse the term "sample mean" here to mean $\breve{\boldsymbol{x}}_{kr}$, instead of $\bar{\boldsymbol{x}}_{kr'}$. In the maximization step, the parameter $\boldsymbol{\mu}_{kr}$ is finally solved by:

$$\boldsymbol{\mu}_{kr} = (\boldsymbol{\Sigma}^{\frac{1}{2}})^t \hat{\boldsymbol{U}} \breve{\boldsymbol{\mu}}^*_{kr} \ .$$

Given the $\boldsymbol{\mu}_{kr}$'s, it is easy to solve $\boldsymbol{\Sigma}$:

$$\boldsymbol{\Sigma} = \frac{\sum_{k=1}^{K} \sum_{r=1}^{R_k} \sum_{i=1}^{n_k} q_{i,kr} (\boldsymbol{x}_i - \boldsymbol{\mu}_{kr})^t (\boldsymbol{x}_i - \boldsymbol{\mu}_{kr})}{n} \ .$$

To initialize the estimation algorithm, we first choose R_k, the number of mixture components for each class k. For simplicity, an equal number of components are assigned to each class. The constrained model is initialized by the estimated parameters from a standard Gaussian mixture model with the same number of components.

We have so far discussed the model estimation in a classification set-up. We assume a common covariance matrix and a common constrained subspace for all the components in all the classes. Similar parameter estimations can also be applied to the clustering model. Specifically, all the data are put in one "class". In this "one-class" estimation problem, all the parameters can be estimated likewise, by omitting the "k" subscript for classes. For brevity, we skip the details here.

4.4.2 Variation of the Algorithm

We have introduced the Gaussian mixture model with component means from different classes constrained in the same subspace. It is natural to modify the

previous constraint in (4.4) to

$$\boldsymbol{v}_j^t \cdot \boldsymbol{\mu}_{k1} = \boldsymbol{v}_j^t \cdot \boldsymbol{\mu}_{k2} = \cdots = \boldsymbol{v}_j^t \cdot \boldsymbol{\mu}_{kR_k} = c_{k,j} \ , \tag{4.10}$$

where $\boldsymbol{v}_j$'s are linearly independent vectors spanning an orthonormal basis, $j = 1, ..., q$, $q < p$, and $c_{k,j}$ depends on class k. That is, the projections of all the component means within class k onto the null space $\boldsymbol{\nu}^{\perp}$ coincide at the constant $\boldsymbol{c}_k$, where $\boldsymbol{c}_k = (c_{k,1}, c_{k,2}..., c_{k,q})^t$. In the new constraint (4.10), $\{\boldsymbol{v}_1, ..., \boldsymbol{v}_q\}$ is the same set of vectors as used in constraint (4.4), which spans the null space $\boldsymbol{\nu}^{\perp}$. Because c_k varies with class k, the subspace in which the component means from each class reside differs from each other by a translation, that is, these subspaces are parallel.

We train a constrained model for each class separately, and assume a common covariance matrix across all the components in all the classes. In the new constraint (4.10), $\boldsymbol{c}_k$ is actually equal to the projection of the class mean $\mathcal{M}'_k$ onto the null space $\boldsymbol{\nu}^{\perp}$. Similar to the previous estimation, in practice, we do not need the value of $\boldsymbol{c}_k$ in the parameter estimation. With the constraint (4.10), essentially, the component means in each class are now constrained in a shifted subspace parallel to $\boldsymbol{\nu}$. The shifting of subspace for each class is determined by $\boldsymbol{c}_k$, or the class mean $\mathcal{M}'_k$. Suppose the dimension of the constrained subspace is d. In general, the dimension that matters for classification in this variation of the algorithm is $d + K - 1$, assuming that the class means already span a subspace of dimension $K - 1$.

We first subtract the class specific means from the data in the training set, that is, do a class specific centering of the data. Similarly as the algorithm outlined in Section 4.4, we put all the centered data from all the classes into one training set, find all the modes under different kernel bandwidths, and then apply MPCA to generate a sequence of constrained subspaces. The reason that we remove the class specific means first is that they have already played a role in spanning the subspace containing all the component means. When applying MPCA, we only want to capture the dominant directions for the variation within the classes.

Comparing with the parameter estimation in Section 4.4, the only change that we need to make is that the constrained q coordinates in $\breve{\boldsymbol{\mu}}^*_{kr}$ are now identical

over r, but not over class k. For the first q coordinates, $j = 1, ..., q$, we have:

$$\breve{\mu}^*_{kr,j} = \frac{\sum_{r'=1}^{R_k} l_{kr'} \breve{x}_{kr',j}}{n_k} , \quad \text{identical over } r \text{ in class } k .$$

That is, the first q constrained coordinates are optimized using component-pooled sample mean in class k. All the other equations in the estimation remain the same.

4.5 Experiments

In this section, we present experimental results on several real and simulated data sets. The mixture model with subspace constrained means, reduced rank MDA, and standard MDA on the projection of data onto the constrained subspace, are compared for the classification of real data with moderate to high dimensions. We also visualize and compare the clustering results of our proposed method and the reduced rank MDA on several simulated data sets.

The detailed methods tested in the experiments and their name abbreviations are summarized as follows:

- **GMM-MPCA** The mixture model with subspace constrained means, in which the subspace is obtained by MPCA.
- **GMM-MPCA-MEAN** The mixture model with subspace constrained means, in which the subspace is obtained by MPCA-MEAN, as introduced in Section 4.3.2.
- **GMM-MPCA-SEP** The mixture model with component means constrained by separately shifted subspace for each class, as introduced in Section 4.4.1.
- **MDA-RR** The reduced rank mixture discriminant analysis (MDA), which is a weighted rank reduction of the full MDA.
- **MDA-RR-OS** The reduced rank mixture discriminant analysis (MDA), which is based on optimal scoring (Hastie and Tibshirani, 1996), a multiple linear regression approach.
- **MDA-DR-MPCA** The standard MDA on the projection of data onto the same constrained subspace selected by GMM-MPCA.

- **MDA-DR-MPCA-MEAN** The standard MDA on the projection of data onto the same constrained subspace selected by GMM-MPCA-MEAN.

Remarks:

1. Since the most relevant work to our proposed method is reduced rank mixture discriminant analysis (MDA), we briefly introduce MDA-RR and MDA-RR-OS in Section 4.5.1.
2. In MDA-DR-MPCA or MDA-DR-MPCA-MEAN, the data are projected onto the constrained subspace which has yielded the largest training likelihood in GMM-MPCA or GMM-MPCA-MEAN. Note that this constrained subspace is spanned by $\boldsymbol{\nu} = \{\boldsymbol{v}_{q+1}, ..., \boldsymbol{v}_p\}$, which is found by MPCA or MPCA-MEAN, rather than the discriminant subspace informative for classification. We then apply standard MDA (assume a common covariance matrix across all the components in all the classes) to the projected training data, and classify the test data projected onto the same subspace. Note that, if we project the data onto the discriminant subspace spanned by $\{\boldsymbol{\Sigma}^{-1}\boldsymbol{v}_j | j = q+1, ..., p\}$, and then apply standard MDA to classification, it is theoretically equivalent to GMM-MPCA or GMM-MPCA-MEAN (ignoring the variation caused by model estimation). The reason that we conduct these comparisons is multi-fold: first, we want to see if there is advantage of the proposed method as compared to a relative naive dimension reduction scheme; second, when the dimension of the data is high, we want to investigate if the proposed method has robust estimation of $\boldsymbol{\Sigma}$; third, we want to investigate the difference between the constrained subspace and the discriminant subspace.

4.5.1 Reduced Rank Mixture Discriminant Analysis

Reduced rank MDA is a data reduction method which allows us to have a low dimensional view on the classification of data in a discriminant subspace, by controlling the within-class spread of component means relative to the between class spread. We outline its estimation method in Appendix B.4, which is a weighted rank reduction of the full mixture solution proposed by Hastie and Tibshirani

(1996). We also show how to obtain the discriminant subspace of the reduced rank method in Appendix B.4.

Hastie and Tibshirani (1996) applied the optimal scoring approach (Breiman and Ihaka, 1984) to fit reduced rank MDA, which converted the discriminant analysis to a nonparametric multiple linear regression problem. By expressing the problem as a multiple regression, the fitting procedures can be generalized using more sophisticated regression methods than linear regression (Hastie and Tibshirani, 1996), for instance, flexible discriminant analysis (FDA) and penalized discriminant analysis (PDA). The use of optimal scoring also has some computational advantages, for instance, using fewer observations than the weighted rank reduction. A software package containing a set of functions to fit MDA, FDA, and PDA by multiple regressions is provided by Hastie and Tibshirani (1996).

Although the above benefits for estimating reduced rank MDA are gained from the optimal scoring approach, there are also some restrictions. For instance, it can not be easily extended to fit a mixture model for clustering since the component means and covariance are not estimated explicitly. In addition, when the dimension of the data is larger than the sample size, optimal scaling can not be used due to the lack of degrees of freedom in regression. In the following experiment section, we will compare our proposed methods with reduced rank MDA. Both our own implementation of reduced rank MDA based on weighted rank reduction of the full mixture, i.e., MDA-RR, and the implementation using optimal scoring from the software package provided by Hastie and Tibshirani (1996), i.e., MDA-RR-OS, are tested.

4.5.2 Classification

Eight data sets from various sources are used for classification. We summarize the detailed information of these data below.

- The **sonar** data set consists of 208 patterns of sonar signals. Each pattern has 60 dimensions and the number of classes is two. The sample sizes of the two classes are $(111, 97)$.
- The **robot** data set has 5456 navigation instances, with 24 dimensions and four classes (826, 2097, 2205, 328).

- The **waveform** data (Hastie et al., 2001) is a simulated three-classes data of 21 features, with a waveform function generating both training and test sets (300, 500).
- The **imagery** semantics data set (Qiao and Li, 2010) contains 1400 images each represented by a 64 dimensional feature vector. These 1400 images come from five classes with different semantics (300, 300, 300, 300, 200).
- The **parkinsons** data set is composed of 195 individual voice recordings, which are of 21 dimensions and divided into two classes (147, 48).
- The **satellite** data set consists of 6435 instances which are square neighborhoods of pixels, with 36 dimensions and six classes (1533, 703, 1358, 626, 707, 1508).
- The **semeion** handwritten digit data have 1593 binary images from ten classes (0-9 digits) with roughly equal sample size in each class. Each image is of 16×16 pixels and thus has 256 dimensions. Four fifths of the images are randomly selected to form a training set and the remaining as testing.
- The **yaleB** face image data (Georghiades et al., 2001; Lee et al., 2005; He et al., 2005) contains gray scale human face images for 38 individuals. Each individual has 64 images, which are of 32×32 pixels, normalized to unit vectors. We randomly select the images of five individuals, and form a data set of 250 training images and 70 test images, with equal sample size for each individual.

The sonar, robot, parkinsons, satellite and semeion data are from the UCI machine learning repository. Among the above data sets, the semeion and yaleB data have high dimensions. The other data sets are of moderately high dimensions.

For the data sets with moderately high dimensions, five-fold cross validation is used to compute their classification accuracy except for the waveform, whose accuracy is the average over ten simulations, the same setting used in (Hastie et al., 2001). We assume a full common covariance matrix across all the components in all the classes. For the semeion and yaleB data sets, the randomly split training and test samples are used to compute their classification accuracy instead of cross validation due to the high computational cost. Since these two data sets are of high dimensions, for all the tested methods, we assume common diagonal covariance

matrices across all the components in all the classes. For simplicity, the same number of mixture components is used to model each class for all the methods.

In our proposed methods, the constrained subspaces are found by MPCA or MPCA-MEAN, introduced in Section 4.3.1 and 4.3.2. Specifically, in MPCA, a sequence of subspaces are identified from the training data by gradually increasing the kernel bandwidth σ_l, i.e., $\sigma_1 < \sigma_2 < \cdots < \sigma_\eta$, $l = 1, 2, ..., \eta$. In practice, we set $\eta = 20$ and choose σ_l's equally spaced from $[0.1\hat{\sigma}, 2\hat{\sigma}]$, where $\hat{\sigma}$ is the largest sample standard deviation of all the dimensions in the data. HMAC is used to obtain the modes at different bandwidths. Note that in HMAC, some σ_l may result in the same clustering as σ_{l-1}, indicating that the bandwidth needs to be increased substantially so that the clustering result will be changed. In our experiments, only the modes at the bandwidth resulting in different clustering from the preceding bandwidth are employed to span the subspace. For the high dimensional data, since the previous kernel bandwidth range $[0.1\hat{\sigma}, 2\hat{\sigma}]$ does not yield a sequence of distinguishable subspaces, we therefore increase their bandwidths. Specifically, for the semeion and yaleB data, the kernel bandwidth σ_l is now chosen equally spaced from $[4\hat{\sigma}, 5\hat{\sigma}]$ and $[2\hat{\sigma}, 3\hat{\sigma}]$, respectively, with the interval being $0.1\hat{\sigma}$. In GMM-MPCA-SEP, since the modes are identified from a new set of class mean removed data, for both the semeion and yaleB data, the kernel bandwidth σ_l is now chosen equally spaced from $[3.1\hat{\sigma}, 5\hat{\sigma}]$, with the interval being $0.1\hat{\sigma}$. For the other data sets, σ_l is still chosen equally spaced from $[0.1\hat{\sigma}, 2\hat{\sigma}]$. In MPCA-MEAN, if the dimension of the constrained subspace is smaller than the class number K, the subspace is obtained by applying weighted PCA only to class means. Otherwise, at each bandwidth, we obtain the subspace by applying weighted PCA to a union set of class means and modes, with 60% weight allocated proportionally to the means and 40% to the modes, that is, $\gamma = 60$. The subspace yielding the largest likelihood on the training data is finally chosen as the constrained subspace.

Classification Results we show the classification results of the tested methods in this section. The classification error rates on data sets of moderately high dimensions are shown in Tables 4.1, 4.2, and 4.3. We vary the discriminant dimension d and also the number of mixture components used for modeling each class. Similarly, Table 4.4 shows the classification error rates on the semeion and yaleB data, which are of high dimensions. For all the methods except GMM-

MPCA-SEP, the dimension of the discriminant subspace equals the dimension of the constrained subspace, denoted by d. For GMM-MPCA-SEP, the dimension of the discriminant space is actually $K-1+d$. In order to compare on a common ground, for GMM-MPCA-SEP, we change the notation for the dimension of the constrained subspace to d', and still denote the dimension of the discriminant subspace by $d = K-1+d'$. The minimum number of dimensions used for classification in GMM-MPCA-SEP is therefore $K-1$. In all these tables, if d is set to be smaller than $K-1$, we do not have the classification results of GMM-MPCA-SEP, which are marked by "NA". In addition, in Table 4.4b, the classification error rates of MDA-RR-OS on yaleB data are not reported since the dimension p of the data is significantly larger than the sample size n. The reduced rank MDA based on optimal scoring approach cannot be applied due to the lack of degree freedom in the regression step for the small n large p problem. The minimum error rate in each column is in bold font. From the results in these tables, we summarize our findings as follows:

- Comparing the three Gaussian mixture models with subspace constrained means, GMM-MPCA-MEAN and GMM-MPCA-SEP usually outperform GMM-MPCA, except on the waveform data. Since the class means are involved in spanning the constrained subspace in GMM-MPCA-MEAN and determine the shifting of the subspace for each class in GMM-MPCA-SEP, the observed advantage of GMM-MPCA-MEAN and GMM-MPCA-SEP indicates that class means are valuable for finding a good subspace.
- Comparing the proposed methods and the reduced rank MDA methods, when the discriminant dimension is low, GMM-MPCA-MEAN and GMM-MPCA-SEP usually perform better than MDA-RR and MDA-RR-OS. When the discriminant dimension becomes higher, we do not observe a clear winner among different methods. The results are very data-dependent. Note that in GMM-MPCA-MEAN, when the discriminant dimension is smaller than $K-1$, the subspace is obtained by applying weighted PCA only to the class means. For most data sets, when the discriminant dimension is very low, GMM-MPCA-MEAN performs best or close to best.
- Comparing the proposed methods and the simple methods of finding the subspace first and then fitting MDA on the data projected onto the subspace,

when the data dimension is moderately high and the discriminant dimension is very low, GMM-MPCA/GMM-MPCA-MEAN usually perform better than MDA-DR-MPCA/MDA-DR-MPCA-MEAN. As the discriminant dimension increases, with certain component numbers, MDA-DR-MPCA/MDA-DR-MPCA-MEAN may have a better classification accuracy. In addition, if the data dimension is very high, for instance, the yaleB data, MDA-DR-MPCA/MDA-DR-MPCA-MEAN may perform better even at lower discriminant dimension. As discussed in Remark 2 of this section, for MDA-DR-MPCA/MDA-DR-MPCA-MEAN and GMM-MPCA/GMM-MPCA-MEAN, we essentially do classification on the data in two different subspaces, i.e., the constrained subspace and the discriminant subspace. For GMM-MPCA/ GMM-MPCA-MEAN, under the subspace constraint, we need to estimate a common covariance matrix, which affects the discriminant subspace, as shown in Section 4.3.3. Generally speaking, when the discriminant dimension becomes higher or the data dimension is high, it becomes more difficult to accurately estimate the covariance matrix. For instance, for the high dimensional data, we assume a common diagonal covariance matrix, so that the covariance estimation becomes feasible and avoids singularity issue. However, this may result in a poor discriminant subspace, which leads to worse classification accuracy. On the other hand, when the data dimension is moderately high and the discriminant dimension is very low, the estimated covariance matrix is more accurate and the discriminant subspace informative for classification is empirically better than the constrained subspace.

- As a final note, when the discriminant dimension is low, MDA-DR-MPCA-MEAN generally outperforms MDA-DR-MPCA.

4.5.3 Sensitivity of Subspace to Bandwidths

Different kernel bandwidths may result in different sets of modes by HMAC, which again may yield different constrained subspaces. We investigate in this section the sensitivity of constrained subspaces to kernel bandwidths.

Assume two subspaces $\boldsymbol{\nu}_1$ and $\boldsymbol{\nu}_2$ are spanned by two sets of orthonormal basis vectors $\{\boldsymbol{v}_1^{(1)}, ..., \boldsymbol{v}_d^{(1)}\}$ and $\{\boldsymbol{v}_1^{(2)}, ..., \boldsymbol{v}_d^{(2)}\}$, where d is the dimension. To

Table 4.1: Classification error rates (%) for the data with moderately high dimensions (I)

(a) Robots data

	Num of components	$d=2$	$d=5$	$d=7$	$d=9$	$d=11$	$d=13$	$d=15$	$d=17$
3	GMM-MPCA	41.39	35.78	31.93	31.73	31.25	31.54	31.65	31.60
	GMM-MPCA-MEAN	**30.32**	32.06	**30.11**	30.52	31.19	31.40	31.69	31.29
	GMM-MPCA-SEP	NA	30.86	30.68	**30.42**	**29.58**	30.28	30.97	30.68
	MDA-RR	41.22	**30.32**	30.85	30.57	29.95	**29.95**	**29.95**	**29.95**
	MDA-RR-OS	40.16	32.73	32.44	30.35	30.66	30.43	30.26	31.01
	MDA-DR-MPCA	44.10	40.30	35.04	32.72	33.03	33.56	33.39	32.84
	MDA-DR-MPCA-MEAN	41.22	36.42	34.71	33.83	32.75	32.44	33.47	32.26
4	GMM-MPCA	40.74	31.91	30.77	30.15	29.40	29.05	27.91	28.24
	GMM-MPCA-MEAN	**26.56**	31.49	**29.71**	29.98	28.43	28.02	28.12	28.39
	GMM-MPCA-SEP	NA	**31.41**	30.19	28.45	28.92	30.13	29.54	30.39
	MDA-RR	40.45	33.63	30.41	**28.28**	**27.77**	**27.09**	**27.18**	**27.18**
	MDA-RR-OS	40.91	31.87	31.36	30.24	27.88	29.01	28.59	28.61
	MDA-DR-MPCA	42.26	36.16	34.64	31.95	30.06	28.90	29.77	27.56
	MDA-DR-MPCA-MEAN	39.41	34.38	34.53	31.96	29.73	29.45	28.15	28.28
5	GMM-MPCA	37.72	29.67	29.25	29.31	27.86	27.91	**26.28**	26.21
	GMM-MPCA-MEAN	**28.72**	27.86	**26.98**	26.69	26.83	**25.90**	26.37	**26.14**
	GMM-MPCA-SEP	NA	**26.48**	27.05	27.46	27.22	26.76	26.74	27.00
	MDA-RR	40.39	29.01	26.52	**26.08**	**26.03**	26.61	26.52	27.09
	MDA-RR-OS	39.96	30.99	29.38	28.24	28.48	27.59	28.24	27.57
	MDA-DR-MPCA	41.07	35.69	32.44	30.86	29.03	27.99	28.52	26.34
	MDA-DR-MPCA-MEAN	38.34	33.10	32.18	30.13	28.56	26.70	27.05	26.28

(b) Waveform data

	Num of components	$d=2$	$d=4$	$d=6$	$d=8$	$d=10$	$d=12$	$d=14$	$d=16$
3	GMM-MPCA	15.70	15.64	16.12	17.10	17.76	17.80	18.24	18.64
	GMM-MPCA-MEAN	16.12	16.14	16.82	17.38	17.76	17.92	17.90	18.84
	GMM-MPCA-SEP	NA	17.08	17.04	17.22	17.44	17.50	17.70	18.34
	MDA-RR	16.00	18.48	18.64	18.58	18.58	18.58	18.58	18.58
	MDA-RR-OS	15.50	17.20	18.14	17.98	18.00	17.84	18.08	17.98
	MDA-DR-MPCA	**14.74**	**15.28**	15.78	**16.14**	**16.58**	17.12	17.62	17.82
	MDA-DR-MPCA-MEAN	**14.74**	15.50	**15.76**	16.50	17.00	**16.94**	**17.26**	**17.48**
4	GMM-MPCA	15.56	16.28	16.06	16.94	17.84	**17.54**	18.58	19.32
	GMM-MPCA-MEAN	15.84	16.70	16.90	17.28	17.96	18.34	18.36	18.84
	GMM-MPCA-SEP	NA	16.34	17.14	17.56	17.56	18.02	18.16	**18.16**
	MDA-RR	15.80	18.12	18.28	19.06	19.26	19.66	19.66	19.66
	MDA-RR-OS	15.50	17.54	18.36	18.36	19.34	18.92	18.72	18.88
	MDA-DR-MPCA	15.18	15.78	**16.00**	**16.36**	17.12	17.64	**17.64**	18.26
	MDA-DR-MPCA-MEAN	**15.12**	**15.86**	16.16	16.70	**17.00**	17.56	17.66	18.40
5	GMM-MPCA	16.44	16.72	16.42	16.96	17.56	17.86	18.66	18.52
	GMM-MPCA-MEAN	16.26	16.30	17.32	17.72	18.04	17.68	18.28	19.04
	GMM-MPCA-SEP	NA	17.24	16.96	17.32	17.40	17.66	17.68	**18.30**
	MDA-RR	16.76	18.18	18.26	19.14	19.16	19.70	19.78	19.78
	MDA-RR-OS	15.80	17.78	18.62	19.02	19.30	18.92	18.92	18.40
	MDA-DR-MPCA	15.34	15.86	**15.98**	16.66	**17.16**	**16.90**	17.90	18.80
	MDA-DR-MPCA-MEAN	**15.08**	**15.70**	16.76	**16.16**	17.30	17.90	**17.56**	18.38

measure the closeness between two subspaces, we project the basis of one subspace onto the other. Specifically, the closeness between $\boldsymbol{\nu}_1$ and $\boldsymbol{\nu}_2$ is defined as $closeness(\boldsymbol{\nu}_1, \boldsymbol{\nu}_2) = \sum_{i=1}^{d}\sum_{j=1}^{d}(\boldsymbol{v}_i^{(1)t} \cdot \boldsymbol{v}_j^{(2)})^2$. If $\boldsymbol{\nu}_1$ and $\boldsymbol{\nu}_2$ span the same subspace, $\sum_{j=1}^{d}(\boldsymbol{v}_i^{(1)t} \cdot \boldsymbol{v}_j^{(2)})^2 = 1$, for $i = 1, 2, ..., d$. If they are orthogonal to each other, $\sum_{j=1}^{d}(\boldsymbol{v}_i^{(1)t} \cdot \boldsymbol{v}_j^{(2)})^2 = 0$, for $i = 1, 2, ..., d$. Therefore, the range of $closeness(\boldsymbol{\nu}_1, \boldsymbol{\nu}_2)$ is $(0, d)$. The higher the value, the closer the two subspaces are.

In our proposed methods, a collection of constrained subspaces are obtained through MPCA or MPCA-MEAN at different kernel bandwidth σ_l's, $l = 1, 2, ..., \eta$, and $\sigma_1 < \sigma_2 < \cdots < \sigma_\eta$. To measure the sensitivity of subspaces to different bandwidths, we compute the mean closeness between the subspace found at σ_l

Table 4.2: Classification error rates (%) for the data with moderately high dimensions (II)

(a) Sonar data

	Num of components	$d=2$	$d=3$	$d=5$	$d=7$	$d=9$	$d=11$	$d=13$	$d=15$
3	GMM-MPCA	39.29	39.78	24.56	25.48	24.51	21.61	21.11	21.13
	GMM-MPCA-MEAN	**35.92**	23.57	23.54	24.04	23.09	22.12	21.63	21.63
	GMM-MPCA-SEP	NA	27.85	25.45	24.54	24.06	24.55	23.56	24.02
	MDA-RR	36.48	28.82	22.08	22.08	22.08	22.08	22.08	22.08
	MDA-RR-OS	45.16	25.87	22.61	24.05	20.68	23.60	22.59	23.58
	MDA-DR-MPCA	42.31	38.45	19.71	**18.33**	19.77	22.18	**20.71**	17.76
	MDA-DR-MPCA-MEAN	39.43	**23.56**	**18.77**	**18.33**	**18.83**	**19.78**	21.20	**16.81**
4	GMM-MPCA	40.53	38.88	**20.19**	20.72	18.32	18.75	17.33	19.74
	GMM-MPCA-MEAN	**35.08**	25.45	20.20	**17.83**	**17.37**	**18.26**	**17.31**	20.71
	GMM-MPCA-SEP	NA	26.51	22.62	22.16	20.25	19.75	20.71	**19.25**
	MDA-RR	46.21	27.91	23.07	19.27	19.27	19.27	19.27	19.27
	MDA-RR-OS	42.80	26.35	26.44	19.23	21.62	22.10	19.25	22.58
	MDA-DR-MPCA	37.50	37.42	22.11	18.33	18.82	21.21	21.23	20.26
	MDA-DR-MPCA-MEAN	40.85	**22.11**	20.24	19.28	20.24	19.76	20.73	19.31
5	GMM-MPCA	44.77	39.78	24.56	25.48	24.51	21.61	21.11	21.13
	GMM-MPCA-MEAN	**35.42**	27.89	**21.15**	19.73	**18.78**	19.71	**18.26**	18.76
	GMM-MPCA-SEP	NA	32.31	29.35	20.21	20.22	20.21	19.23	21.17
	MDA-RR	43.70	27.38	25.91	22.06	19.67	**19.67**	19.67	19.67
	MDA-RR-OS	35.55	29.34	24.86	22.12	20.68	22.19	21.18	20.17
	MDA-DR-MPCA	36.05	35.07	21.20	19.29	20.73	23.09	20.71	18.18
	MDA-DR-MPCA-MEAN	38.37	**26.44**	21.21	**18.34**	23.10	24.56	21.64	**18.74**

(b) Imagery data

	Num of components	$d=2$	$d=4$	$d=6$	$d=8$	$d=10$	$d=12$	$d=14$	$d=16$
3	GMM-MPCA	55.36	48.00	40.36	38.64	38.36	37.43	36.07	37.86
	GMM-MPCA-MEAN	**44.50**	**36.21**	36.86	37.07	36.36	36.79	36.71	36.14
	GMM-MPCA-SEP	NA	NA	**35.21**	**34.07**	35.57	35.79	35.14	35.64
	MDA-RR	52.57	43.14	40.21	35.86	35.86	35.71	35.29	35.29
	MDA-RR-OS	52.36	42.50	38.50	**34.07**	**35.29**	**35.50**	**34.93**	**34.79**
	MDA-DR-MPCA	59.93	49.36	42.21	41.71	41.00	39.50	37.00	38.14
	MDA-DR-MPCA-MEAN	49.36	44.14	40.86	40.93	38.64	38.50	37.79	38.07
4	GMM-MPCA	57.00	48.29	39.79	38.14	36.57	36.93	35.64	36.64
	GMM-MPCA-MEAN	**45.00**	**37.00**	39.21	36.57	35.36	35.43	35.86	36.14
	GMM-MPCA-SEP	NA	NA	**35.00**	**35.43**	35.07	35.50	35.43	35.00
	MDA-RR	52.21	40.64	38.93	35.79	37.50	36.50	35.29	34.86
	MDA-RR-OS	51.64	43.57	37.64	35.50	**34.50**	**32.36**	**34.50**	**33.64**
	MDA-DR-MPCA	59.71	50.00	40.14	40.36	38.29	37.86	36.29	37.64
	MDA-DR-MPCA-MEAN	49.71	42.36	39.71	39.71	38.64	37.43	37.21	37.93
5	GMM-MPCA	57.79	48.50	40.36	37.57	37.36	39.07	36.07	38.29
	GMM-MPCA-MEAN	**45.64**	**36.57**	38.64	36.14	37.00	36.64	35.64	35.36
	GMM-MPCA-SEP	NA	NA	**35.79**	35.14	34.43	34.36	35.57	35.43
	MDA-RR	53.21	43.36	39.00	36.07	35.86	34.43	33.93	34.07
	MDA-RR-OS	52.07	42.57	39.71	**34.21**	**32.64**	**34.21**	**33.50**	**32.93**
	MDA-DR-MPCA	58.50	48.93	39.79	38.21	39.57	39.07	36.00	38.71
	MDA-DR-MPCA-MEAN	50.00	42.86	39.21	38.36	39.00	37.57	37.64	36.21

and all the other subspaces at preceding bandwidths $\sigma_{l'}, l' = 1, 2, ..., l-1$. A large mean closeness indicates that the current subspace is close to preceding subspaces. Table 4.5 lists the mean closeness of subspaces by MPCA and MPCA-MEAN at different bandwidth levels for the sonar and imagery data (the training set from one fold in the previous five-fold cross validation setup). The first values in the parentheses are by MPCA while the second values are by MPCA-MEAN. We vary the dimension of the constrained subspace. The number of modes identified at each level is also shown in the tables. As Table 4.5 shows, for both methods, the subspaces found at the first few levels are close to each other, indicated by their large mean closeness values, which are close to d, the dimension of the subspace.

Table 4.3: Classification error rates (%) for the data with moderately high dimensions (III)

(a) Parkinsons data

	Num of components	$d=2$	$d=3$	$d=5$	$d=7$	$d=9$	$d=11$	$d=13$	$d=15$
3	GMM-MPCA	17.96	17.42	14.33	14.84	16.98	14.92	15.47	12.84
	GMM-MPCA-MEAN	18.90	14.84	**11.75**	13.33	13.88	**12.88**	13.89	12.85
	GMM-MPCA-SEP	NA	**11.75**	13.29	**12.26**	13.34	13.85	**11.25**	13.34
	MDA-RR	19.42	15.96	13.88	13.88	13.88	13.88	13.88	13.88
	MDA-RR-OS	**16.88**	16.42	12.31	13.89	**12.31**	13.89	13.37	**12.31**
	MDA-DR-MPCA	19.47	17.90	13.81	14.35	15.37	14.83	15.38	16.41
	MDA-DR-MPCA-MEAN	19.47	17.90	13.81	14.33	15.37	15.35	15.35	15.35
4	GMM-MPCA	17.88	14.77	14.31	14.81	12.84	11.30	10.28	12.32
	GMM-MPCA-MEAN	**14.81**	14.31	13.83	12.81	**9.25**	**9.28**	**8.74**	10.76
	GMM-MPCA-SEP	NA	**11.28**	11.33	12.29	11.80	10.29	9.76	**9.79**
	MDA-RR	16.85	12.81	11.79	**10.29**	9.79	9.79	9.79	**9.79**
	MDA-RR-OS	18.41	15.38	**10.74**	10.79	11.84	11.83	12.85	10.32
	MDA-DR-MPCA	19.47	18.47	12.29	12.35	10.77	11.23	10.72	12.30
	MDA-DR-MPCA-MEAN	19.47	17.43	12.29	11.81	9.72	10.24	10.72	11.76
5	GMM-MPCA	19.39	18.39	14.84	17.37	15.31	12.81	11.25	12.79
	GMM-MPCA-MEAN	**18.39**	16.34	13.26	14.83	11.78	11.25	10.25	11.29
	GMM-MPCA-SEP	NA	**14.81**	**10.75**	12.25	11.75	**10.75**	10.25	10.78
	MDA-RR	19.94	16.30	12.27	13.83	11.28	10.78	11.28	10.79
	MDA-RR-OS	18.96	16.43	14.30	**11.22**	10.25	12.33	**9.71**	**9.74**
	MDA-DR-MPCA	18.96	18.47	13.80	11.81	11.28	12.77	10.70	10.76
	MDA-DR-MPCA-MEAN	18.96	19.52	12.26	11.31	**9.75**	12.27	10.20	**9.74**

(b) Satellite data

	Num of components	$d=2$	$d=4$	$d=7$	$d=9$	$d=11$	$d=13$	$d=15$	$d=17$
3	GMM-MPCA	**16.74**	15.01	14.16	14.67	14.06	13.95	13.97	13.63
	GMM-MPCA-MEAN	16.94	14.10	13.53	13.77	13.95	13.78	13.66	13.68
	GMM-MPCA-SEP	NA	NA	15.48	13.58	13.80	13.58	13.71	13.67
	MDA-RR	35.18	14.41	12.84	**12.96**	13.46	13.60	13.66	13.53
	MDA-RR-OS	34.90	**13.95**	**13.01**	13.09	**12.82**	**13.04**	**13.35**	13.29
	MDA-DR-MPCA	17.20	14.83	13.61	13.91	13.80	13.35	13.60	13.69
	MDA-DR-MPCA-MEAN	17.09	14.42	13.58	14.12	14.06	13.41	13.38	**13.13**
4	GMM-MPCA	**17.02**	14.13	13.61	13.80	13.58	12.90	12.93	12.88
	GMM-MPCA-MEAN	17.31	13.41	13.50	13.53	13.24	12.94	12.91	12.87
	GMM-MPCA-SEP	NA	NA	15.40	12.93	13.08	13.35	13.38	13.54
	MDA-RR	35.06	**13.35**	12.60	12.77	12.74	12.73	12.63	13.05
	MDA-RR-OS	34.28	13.49	**11.95**	**12.17**	**11.90**	12.49	**11.97**	**12.14**
	MDA-DR-MPCA	17.54	14.14	13.21	13.52	13.05	12.74	12.46	12.45
	MDA-DR-MPCA-MEAN	17.37	13.36	13.53	13.57	13.07	**12.43**	12.45	12.82
5	GMM-MPCA	**16.25**	13.66	12.90	13.29	12.79	12.26	11.92	12.24
	GMM-MPCA-MEAN	16.77	12.93	12.85	12.96	12.40	12.18	11.89	12.24
	GMM-MPCA-SEP	NA	NA	15.48	13.21	12.56	12.70	12.59	12.49
	MDA-RR	27.43	13.27	12.85	12.34	12.15	12.18	12.29	12.28
	MDA-RR-OS	30.16	13.30	**12.31**	**12.23**	**11.73**	**11.89**	11.98	**11.97**
	MDA-DR-MPCA	16.61	13.80	12.70	12.82	12.74	12.09	**11.79**	12.42
	MDA-DR-MPCA-MEAN	16.58	**12.82**	12.99	12.93	12.66	11.92	11.92	12.31

As the bandwidth σ_l increases, the mean closeness starts to decline, which indicates that the corresponding subspace changes. When σ_l is small, the number of modes identified by HMAC is large. The modes and their associated weights do not change much. As a result, the generated subspaces at these bandwidths are relatively stable. As σ_l increases, the kernel density estimate becomes smoother, and more data points tend to ascend to the same mode. We thus have a smaller number of modes with changing weights, which may yield a substantially different subspace. Additionally, the subspace by MPCA-MEAN is spanned by applying weighted PCA to a union set of modes and class means. In our experiment, we have allocated

Table 4.4: Classification error rates (%) for the data with high dimensions

(a) Semeion data

	Num of components	$d=2$	$d=4$	$d=8$	$d=11$	$d=13$	$d=15$	$d=17$	$d=19$
3	GMM-MPCA	53.56	29.72	18.27	19.81	18.89	19.20	18.89	18.27
	GMM-MPCA-MEAN	49.54	29.10	13.31	14.86	16.72	14.86	16.72	16.10
	GMM-MPCA-SEP	NA	NA	NA	13.31	12.07	13.00	16.10	14.86
	MDA-RR	**45.51**	26.93	15.79	14.86	13.93	15.79	14.24	13.62
	MDA-RR-OS	48.36	**24.92**	13.93	**12.41**	**10.60**	**11.10**	**10.59**	**11.41**
	MDA-DR-MPCA	49.54	27.86	19.20	17.03	17.03	15.79	16.72	16.41
	MDA-DR-MPCA-MEAN	48.30	26.01	**12.07**	13.62	13.31	12.07	14.24	14.55
4	GMM-MPCA	53.56	26.32	17.03	16.10	16.10	16.10	16.10	15.17
	GMM-MPCA-MEAN	51.39	25.70	**11.46**	11.76	12.69	13.00	14.24	15.17
	GMM-MPCA-SEP	NA	NA	NA	13.31	12.07	12.07	13.62	11.76
	MDA-RR	49.23	26.01	13.93	13.62	13.00	12.07	13.62	12.07
	MDA-RR-OS	48.70	**24.83**	14.21	**11.60**	**10.59**	**11.16**	**9.97**	**11.09**
	MDA-DR-MPCA	46.75	26.32	17.34	16.41	16.41	15.17	16.10	15.17
	MDA-DR-MPCA-MEAN	**44.58**	26.32	13.00	11.76	15.17	13.31	13.00	13.00
5	GMM-MPCA	51.70	**24.46**	15.79	13.62	15.17	15.17	13.93	13.00
	GMM-MPCA-MEAN	**43.03**	26.63	11.15	11.46	12.38	12.07	13.31	13.00
	GMM-MPCA-SEP	NA	NA	NA	13.00	11.46	13.00	12.38	13.00
	MDA-RR	48.92	25.39	13.00	12.69	11.46	**10.53**	12.07	12.69
	MDA-RR-OS	49.16	26.53	14.21	11.10	**10.60**	**10.53**	9.84	9.96
	MDA-DR-MPCA	48.61	27.24	18.58	13.93	14.24	13.62	13.93	10.84
	MDA-DR-MPCA-MEAN	46.13	25.08	**10.22**	**10.84**	10.84	10.53	**8.98**	**9.29**

(b) YaleB data

	Num of components	$d=2$	$d=4$	$d=6$	$d=8$	$d=10$	$d=12$	$d=14$	$d=16$
3	GMM-MPCA	84.29	64.29	64.29	55.71	45.71	38.57	40.00	34.29
	GMM-MPCA-MEAN	31.43	**17.14**	52.86	51.43	38.57	30.00	28.57	27.14
	GMM-MPCA-SEP	NA	NA	**27.14**	20.00	**21.43**	20.00	20.00	20.00
	MDA-RR	87.14	42.86	**27.14**	**17.14**	28.57	**8.57**	**11.43**	**11.43**
	MDA-DR-MPCA	82.86	58.57	50.00	44.29	37.14	42.86	25.71	32.86
	MDA-DR-MPCA-MEAN	**30.00**	**17.14**	60.00	37.14	40.00	21.43	17.14	14.29
4	GMM-MPCA	84.29	67.14	68.57	55.71	44.29	44.29	40.00	37.14
	GMM-MPCA-MEAN	34.29	22.86	64.29	50.00	35.71	30.00	35.71	30.00
	GMM-MPCA-SEP	NA	NA	**31.43**	25.71	28.57	27.14	24.29	25.71
	MDA-RR	85.71	60.00	41.43	**24.29**	**14.29**	**10.00**	12.86	**11.43**
	MDA-DR-MPCA	90.00	55.71	50.00	42.86	37.14	41.43	28.57	27.14
	MDA-DR-MPCA-MEAN	**25.71**	**21.43**	60.00	35.71	32.86	11.43	**11.43**	12.86
5	GMM-MPCA	85.71	65.71	65.71	55.71	50.00	45.71	42.86	40.00
	GMM-MPCA-MEAN	37.14	**14.29**	60.00	51.43	47.14	42.86	41.43	38.57
	GMM-MPCA-SEP	NA	NA	**31.43**	35.71	**30.00**	32.86	28.57	35.71
	MDA-RR	85.71	61.43	42.86	42.86	32.86	30.00	22.86	24.29
	MDA-DR-MPCA	87.14	67.14	52.86	38.57	34.29	34.29	**17.14**	22.86
	MDA-DR-MPCA-MEAN	**27.14**	18.57	50.00	**34.29**	**27.14**	**21.43**	20.00	**7.14**

a larger weight proportionally to class means (in total, 60%) and the class means remain unchanged in the union set at each kernel bandwidth. Therefore, the differences between subspaces by MPCA-MEAN are smaller than that by MPCA, indicated by larger closeness values.

4.5.4 Model Selection

In our proposed method, the following model selection strategy is adopted. We take a sequence of subspaces resulting from different kernel bandwidths, and then estimate a mixture model constrained by each subspace and finally choose a model yielding the maximum likelihood. In this section, we examine our model selection criteria, and the relationships among test classification error rates, training

likelihoods and kernel bandwidths.

Figure 4.1 shows the test classification error rates at different levels of kernel bandwidth for several data sets (from one fold in the previous five-fold cross validation setup), when the number of mixture components for each class is set to three. The error rates are close to each other at the first few levels. As the kernel bandwidth increases, the error rates start to change. Except for the waveform, on which the error rates of GMM-MPCA and GMM-MPCA-MEAN are very close, for the other data sets in Figure 4.1, the error rate of GMM-MPCA-MEAN at each bandwidth level is lower than that of GMM-MPCA. Similarly, at each kernel bandwidth level, the error rate of GMM-MPCA-SEP is also lower than that of GMM-MPCA, except for the robot data. We also show the training log-likelihoods of these methods with respect to different kernel bandwidth levels in Figure 4.2. The training log-likelihoods are also stable at the first few levels and start to fluctuate as the bandwidth increases. This is due to the possible big change in subspaces under large kernel bandwidths.

In our model selection strategy, the subspace which results in the maximum log likelihood of the training model is selected and then we apply the model under the constraint of that specific subspace to classify the test data. In Figure 4.1, the test error rate of the model which has the largest training likelihood is indicated by an arrow. As we can see, for each method, this error rate is mostly ranked in the middle among all the error rates at different levels of bandwidth, which indicate that our model selection strategy helps find a reasonable training model.

Table 4.5: Mean closeness of subspaces by MPCA and MPCA-MEAN at different levels of kernel bandwidth

(a) Sonar data

Bandwidth level	2	4	6	8	10	12	14
Num of modes	158	144	114	86	60	15	8
$d = 2$	(2.00, 2.00)	(2.00, 2.00)	(2.00, 2.00)	(1.99, 1.99)	(1.98, 1.98)	(1.77, 1.88)	(1.43, 1.80)
$d = 4$	(4.00, 4.00)	(4.00, 4.00)	(3.99, 4.00)	(3.98, 3.99)	(3.84, 3.94)	(2.63, 3.08)	(2.09, 2.96)
$d = 6$	(6.00, 6.00)	(5.99, 5.99)	(5.95, 5.99)	(5.91, 5.88)	(5.41, 5.47)	(4.48, 4.64)	(3.48, 4.13)
$d = 8$	(8.00, 8.00)	(8.00, 8.00)	(7.97, 7.97)	(7.86, 7.89)	(6.98, 7.00)	(6.20, 6.40)	(4.29, 5.18)

(b) Imagery data

Bandwidth level	2	4	6	8	10	12
Num of modes	1109	746	343	144	60	35
$d = 6$	(6.00, 6.00)	(5.97, 5.99)	(5.32, 5.86)	(5.34, 5.61)	(5.21, 5.32)	(5.02, 5.32)
$d = 8$	(8.00, 8.000)	(7.96, 7.96)	(7.54, 7.89)	(7.31, 7.76)	(6.82, 7.43)	(6.27, 6.85)
$d = 10$	(10.00, 10.00)	(9.80, 9.52)	(9.56, 9.48)	(9.25, 9.23)	(8.45, 9.01)	(7.71, 8.42)
$d = 12$	(12.00, 12.00)	(11.96, 11.96)	(11.48, 11.50)	(10.29, 10.89)	(10.06, 10.33)	(9.49, 9.87)

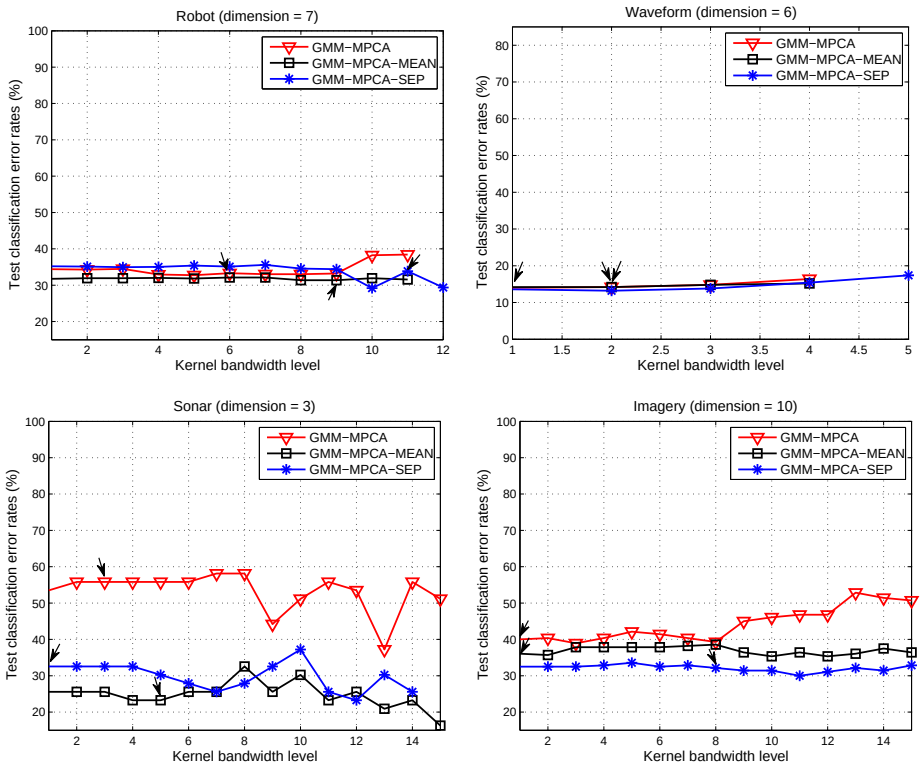


Figure 4.1: The test classification error rates at different levels of kernel bandwidth

4.5.5 Clustering

We present the clustering results of GMM-MPCA and MDA-RR on several simulated data sets and visualize the results in a low-dimensional subspace. The previous model selection criteria is also used in clustering. After fitting a subspace constrained Gaussian mixture model, all the data points having the highest posterior probability belonging to a particular component form one cluster. We outline the data simulation process as follows.

The data is generated from some existing subspace constrained model. Specifically, we take the training set of the imagery data from one fold in the previous five-fold cross validation setup and estimate its distribution by fitting a mixture model using GMM-MPCA. We will obtain five estimated component means which are ensured to be constrained in a two dimensional subspace. A set of 200 samples are randomly drawn from a multivariate Gaussian distribution with the previously estimated component means as the sample means. A common identity covariance

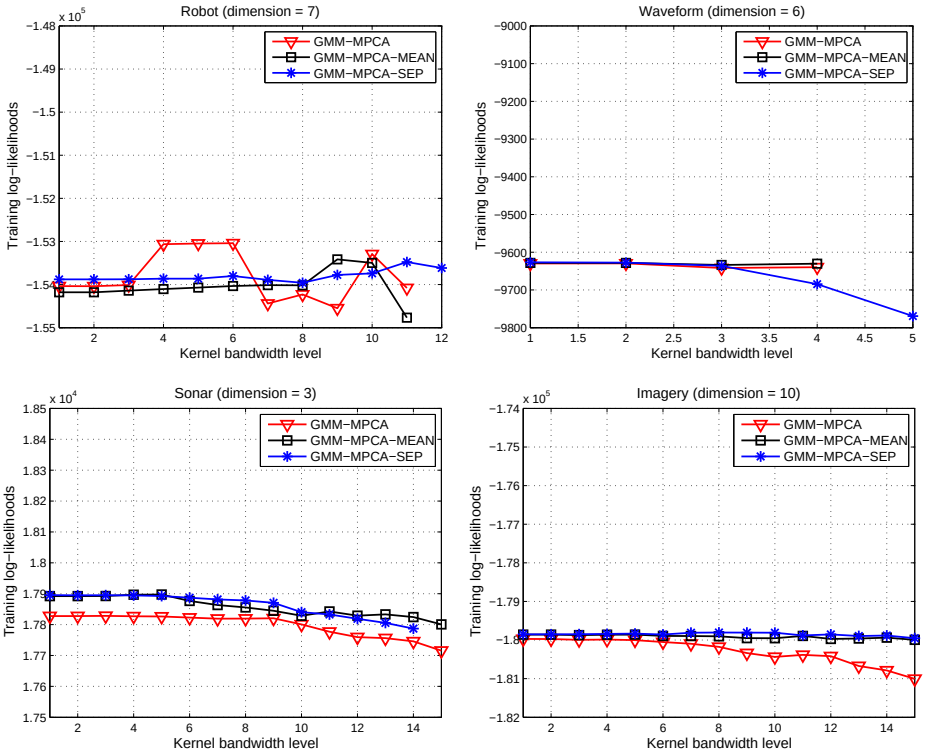


Figure 4.2: Training log-likelihoods at different kernel bandwidth levels

is assumed for the Gaussian multivariate distributions. We generate five sets of samples in this way, forming a data set of 1000 samples. We scale the component means by different factors so that the data have different levels of dispersion among the clusters. The lower the dispersion, the more difficult the clustering task. Specifically, the scaling factor is set to be 0.125, 0.150, and 0.250, respectively, generating three simulated data with low, middle and high level dispersion between clusters.

Figure 4.3 shows the clustering results of three simulated data sets by GMM-MPCA and MDM-RR, in two-dimensional plots, color-coding the clusters. The data projected onto the true discriminant subspace with true cluster labels are shown in Figure 4.3(a). In addition, Figure 4.3(b) and Figure 4.3(c) show the data projected onto the two-dimensional discriminant subspaces by GMM-MPCA and MDA-RR. For all the simulated data sets, both GMM-MPCA and MDA-RR can effectively reveal the clustering structure in a low-dimensional subspace. To evaluate their clustering performance, we compute the clustering accuracy by

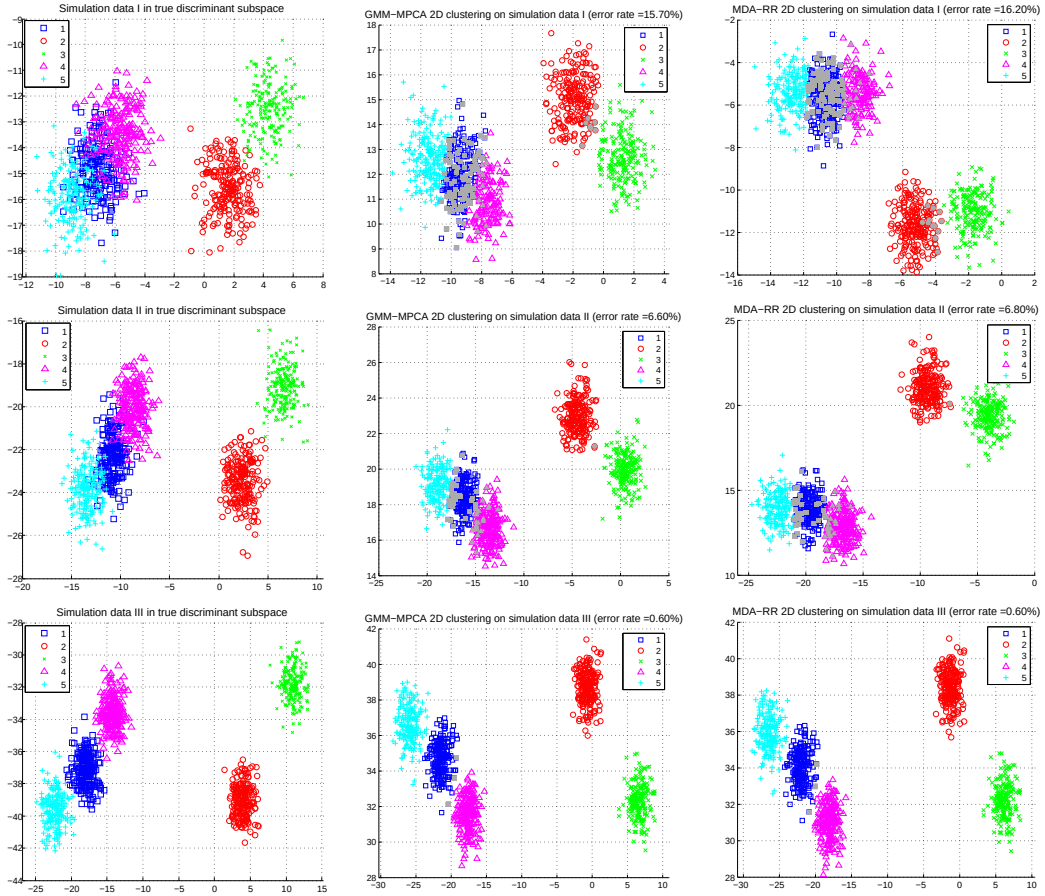
Table 4.6: Closeness between subspaces in clustering with different dispersions

Closeness	low	middle	high
GMM-MPCA	1.769	1.820	1.881
MDA-RR	1.552	1.760	1.866

comparing their predicted and true clustering labels. Suppose the true cluster label of data point $\boldsymbol{x}_i$ is t_i and the predicted cluster label is p_i, the clustering error rate is calculated as $1-\sum_{i=1}^{n} I(t_i, map(p_i))/n$, where n is the total number of data points, $I(x, y)$ is an indicator function that is equal to one if $x = y$ otherwise zero, and $map(p_i)$ is a permutation function which maps the predicted label to an equivalent label in the data set. Specifically, we use the Kuhn-Munkres algorithm to find the best matching (Lovász and Plummer, 1986). The clustering error rates are listed in the titles above the plots in Figure 4.3. The mis-clustered data points are in gray. When the dispersion between clusters is low or middle, the clustering error rates of GMM-MPCA are smaller than those of MDA-RR. When the dispersion is high, the task becomes relatively easy and the clustering accuracy of these two methods are the same. In Table 4.6, we also show the closeness between the true discriminant subspace and the discriminant subspaces found by GMM-MPCA and MDA-RR. Comparing with MDA-RR, for all the three data sets, the closeness between the subspace by GMM-MPCA and the true subspace are smaller.

4.6 Summary

In this chapter, we propose a Gaussian mixture model with the component means constrained in a pre-selected subspace. We prove that the modes, the component means of a Gaussian mixture, and the class means all lie in the same constrained subspace. Several approaches to finding the subspace are proposed by applying weighted PCA to the modes, class means, or a union set of modes and class means. The constrained method results in a dimension reduction property, which allows us to view the classification or clustering structure of the data in a lower dimensional space. An EM-type algorithm is derived to estimate the model, given any constrained subspace. In addition, the Gaussian mixture model with the component means constrained by separate parallel subspace for each class is investigated. Although reduced rank MDA is a competitive classification method by constraining



(a) (b) (c)

Figure 4.3: Two-dimensional plot for the clustering of synthetic data, color-coding the clusters. (a) Projections onto the two-dimensional true discriminant subspace, with true cluster labels. (b), (c) Projections onto the two-dimensional discriminant subspace by GMM-MPCA and MDA-RR respectively, with predicted cluster labels.

the class means to an optimal discriminant subspace within each EM iteration, experiments on several real data sets of moderate to high dimensions show that when the dimension of the discriminant subspace is very low, it is often outperformed by our proposed method with a simple technique of spanning the constrained subspace using only class means.

We select the constrained subspace which has the largest training likelihood among a sequence of subspaces resulting from different kernel bandwidths. If the number of candidate subspaces is large, it may be desired to narrow down the

search by incorporating some prior knowledge. For instance, the proposed method may have a potential in visualization when users already know that only a certain dimensions of the data matter for classification or clustering, i.e., a constrained subspace can be obtained beforehand. Finally, we expect this subspace constrained method can be extended to other parametric mixtures, for instance, mixture of Poisson for discrete data.

Chapter 5

Parallel Hierarchical Mode Association Clustering

5.1 Introduction

Hierarchical Model Association Clustering (HMAC) is a nonparametric clustering method which groups data points into one cluster if they are associated with the same mode in a mixture density (Li et al., 2007). In Chapter 4, we introduce a Gaussian mixture model with component means constrained in pre-selected subspaces. The subspace is found by applying weighted principal component analysis to the modes of a kernel density and the class means. Note that the modes are obtained by HMAC. HMAC is robust in high dimensions, especially when clusters of data deviate from Gaussian distributions. HMAC has been applied to segment images for the analysis of color combination aesthetics (Yao et al., 2012). All the modal vectors are extracted as the representative colors of an image. Since these modal vectors are the local maximums of a mixture density, specifically, a kernel density estimator, the representative colors tend to be more "bright" and thus better retain the true colors. We have also applied HMAC to perform clustering on industry engineering design data in work-centered visual analytics to aid the search for optimal designs (Yan et tal., 2012a). As the computational component in a visual analytics system, HMAC groups similar design data into clusters and reveals the underlying patterns. The clustering results are passed to a visualiza-

tion component which interacts with human by inputting selected data for the computational component and outputting visualization results.

One challenge faced by HMAC and any other clustering methods is the scalability issue with large scale data sets. In many applications, the clustering results have to be rendered within a very short time. For instance, in visual analytics of large scale multi-dimensional engineering design data, the capability of real-time human and computer interaction is critical (Yan et tal., 2012b). While the visualization component can display results very fast, the computational component usually takes much longer time to mine data, find patterns and finally generate results. We thus need a fast computational algorithm that can have the results readily available within a very short time, bridging the gap between visualization and data mining. Another example is with the image segmentation using HMAC. If we aim to provide on-site color aesthetic feedback of high resolution photos to mobile camera consumers, the segmentation process have to be finished instantly, in addition to the data sending, receiving and analyzing cost.

In this chapter, we introduce two parallel versions of HMAC which can dramatically reduce the computational time of the original algorithm using parallel computing. The basic idea of parallel computing is to either partition the core task into various tasks that can be performed independently (task parallelism), or partition the data into pieces which can be dealt with separately (data parallelism). The final result or solution is obtained by combining the partial results when all the sub-problems are solved (Pacheco, 2011). Two most popular parallel computing frameworks are MapReduce (Dean, 2008) and message passing interface (MPI). MapReduce has two steps: a "map" step to partition data into subsets that can be processed independently on each compute node and a "reduce" step to merge the results from the previous step. It serves as an API of the parallel computing framework. Users do not have to worry about the partition of the data, scheduling of tasks, communications, and logging. All these are handled internally by the framework. Hadoop is an open source platform for providing the MapReduce functions. MapReduce offers a user friendly approach to implementing parallel algorithms. MPI is a standardized portable message passing library (Gropp et tal., 1999), offering programmers explicit control over how machines send data or messages to each other. Comparing with MapReduce, MPI is more demanding

yet more flexible. In real practice, we found that MapReduce algorithms are more suitable for data-intensive problems with non-iterative procedures and little data exchange, while MPI algorithms are appropriate to computational-intensive problems involving iterations and huge data exchange. This is also similar to what have been found in (Chen, 2011). Since HMAC has iterative process and is computationally intensive, MPI framework is more appropriate.

According to the inference of HMAC (Li et al., 2007), the most computationally intensive step is the modal Expectation-Maximization (MEM) algorithm which finds the density mode for each data point using an iterative optimization. To speed up HMAC, we have designed two parallel approaches. The first approach is to parallelize each iterative step of MEM by dividing the data into several subsets and assigning the calculation related with each subset data to a single compute node (or a slave node). The computation is performed simultaneously on each node. All the partial results are finally summarized by a master node. In the second approach, we still divide the data into several subsets and assign each subset to a single compute node. However, instead of parallelizing each MEM step in HMAC, we have each compute node calculate the density modes for the assigned data points using the original MEM. For both approaches, a master node finally merges the found modes if they are numerically close. All the data points that are associated with the same mode form one cluster. In the following sections, we will provide details about these two parallel approaches.

5.2 Parallel Approach I

We parallelize the MEM algorithm, which finds a density mode for each data point. MEM comprises two iterative steps, similar to the expectation and maximization steps in EM (Dempster et al., 1977). Let a mixture density be

$$f(x) = \sum_{k=1}^{K} \pi_k f_k(x) \ , \tag{5.1}$$

where $x \in \mathcal{R}^d$, π_k is the prior probability of mixture component k, and $f_k(x)$ is the density of component k. Given any initial value $x^{(0)}$, MEM solves a local maximum

of the mixture density, i.e., the mode, by alternating the following two steps until a stopping criterion is met. Start with $t = 0$,

1. E-step: $p_r = \frac{\pi_r f_r(\boldsymbol{x}^{(t)})}{f(\boldsymbol{x}^{(t)})}$, $r = 1, ..., R$.
2. M-step: $\boldsymbol{x}^{(t+1)} = \text{argmax}_{\boldsymbol{x}} \sum_{r=1}^{R} p_r \log f_r(\boldsymbol{x})$.

If the mixture density is a mixture of Gaussians with common covariance matrix, i.e., $f_k(x) = \phi(x|\mu_k, \Sigma)$, where $\phi(\cdot)$ is the probability density function of a Gaussian distribution, we simply have $x^{(t+1)} = \sum_{k=1}^{K} p_k \mu_k$ in the M step. In the special case of a Gaussian kernel density function, $\mu_k = x_k$, $\pi_k = 1/n$, $K = n$, where n is the total number of data points. In Gaussian kernel, we use a spherical covariance matrix $\Sigma = diag(\sigma^2, \sigma^2, ..., \sigma^2)$, where the standard deviation σ is referred to as the bandwidth of the kernel. Let us denote $diag(\sigma^2, \sigma^2, ..., \sigma^2)$ by $D(\sigma^2)$ for brevity. In HMAC, we model data using Gaussian kernel density, i.e., $f_k(x) = \phi(x|x_k, D(\sigma^2))$. Therefore, the above two steps are simplified as

1. E-step: $p_k = \frac{\phi(x^{(t)}|x_k, D(\sigma^2))}{\sum_{k=1}^{n} \phi(x^{(t)}|x_k, D(\sigma^2))}$, $k = 1, ..., n$.
2. M-step: $x^{(t+1)} = \sum_{k=1}^{n} p_k x_k$.

Now we describe the parallel approach. Suppose the total number of slave nodes is T. Denote the subset of data assigned to a slave node i by S_i. In the E step, for each p_k, the slave node processes its own assigned subset of data. Specifically, we have slave node i compute $\sum_{k \in S_i} \phi(x^{(t)}|x_k, D(\sigma^2))$. The master node will then divide $\phi(x^{(t)}|x_k, D(\sigma^2))$ by the sum of all the partial results from the slave nodes. Similarly, in the M-step, we have each slave node i compute $\sum_{k \in S_i} p_k x_k$ and the master node will sum up all the partial results. After the modes for all the data points are obtained, the master node will merge them (if some modes are numerically close, they will be grouped as a single mode) and the data points that ascend to the same mode will form a cluster.

5.3 Parallel Approach II

Comparing with the first approach, our second parallel approach is simpler and more straightforward. Suppose we have T slave nodes. As shown in Figure 5.1, we

assign a subset of data to each slave node i, denoted by S_i. Instead of parallelizing the MEM algorithm, we have each node find the modes for the assigned subset of data using the original MEM. In this way, there is no data transferring operation involved in the iterative MEM. Note that, to find the mode for each data point, we still need the entire data in the MEM computation. Therefore, each slave node should be able to access all the data. Denote the modes found by slave node i for the assigned subset S_i by M_i. Similar to the first approach, a master node will merge the modes $\{M_1, M_2, ..., M_t\}$ if some are numerically close and then form clusters by grouping the data points that are associated with the same mode.

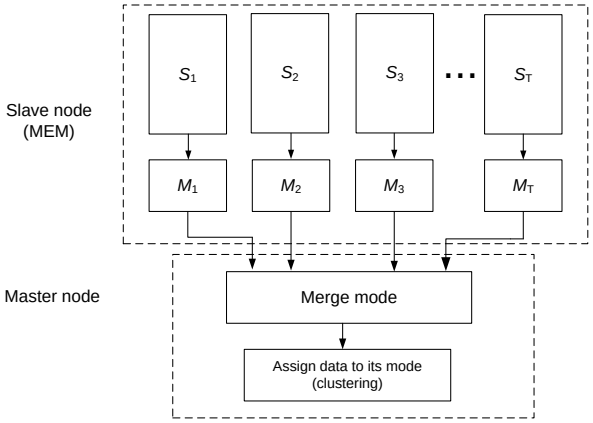


Figure 5.1: The flow of parallel approach II

5.4 Remarks on Parallel HMAC

HMAC is a hierarchical clustering algorithm. When the kernel bandwidth σ increases, the kernel density estimate becomes smoother and more data points tend to climb to the same model. Give a sequence of bandwidths $\sigma_1 < \sigma_2 < ... < \sigma_\eta$, hierarchical clustering is performed in a bottom-up manner. As introduced in Section 4.2, HMAC starts with every point being a cluster by itself, which corresponds to the extreme case where σ_1 approaches 0. At any bandwidth $\sigma_l (l > 1)$, the modes, that is, cluster representatives, obtained from the preceding bandwidth are input to the modal EM algorithm. The modes identified then form a new set of cluster representatives. This procedure is repeated across all σ_l's. For details, we refer interested readers to (Li et al., 2007). The hierarchical clustering results

are thus nested, as a dendrogram. The proposed parallel approaches thus have to be applied to the data that need to be clustered on each level of the hierarchy. HMAC also provides users the option of not doing nested hierarchy. In that case, a collection of clustering results can be obtained using a sequence of kernel bandwidths. As the bandwidth increases, the number of clusters will decrease. Since the clustering of data under each bandwidth is independent, this non-nested clustering process can be readily parallelized as well.

5.5 Experiments

The performance of these two proposed parallel HMAC approaches using MPI are reported in this section. We apply the parallel algorithms on the ship design data used in the visual analytic system (Yan et tal., 2012b), which has 2,000 samples and 17 dimensions, and the imagery data (Qiao and Li, 2010) which has 1,400 samples and 64 dimensions. The default parameters of original HMAC are used [1]. Figure 5.2 shows the running time of these two parallel approaches when the number of compute nodes increases (the number of master node is always one and the remaining are slave nodes).

These two parallel approaches are implemented using the OpenMPI library. We run the experiments on the CyberSTAR cluster computing platform at Penn State. Each compute node has an Intel Xeon processor with 2.67GHZ. In Figure 5.2, the two parallel approaches using MPI are denoted by "MPI-P1" and "MPI-P2", respectively. As we can see, as the number of compute nodes increases, for both data sets, the running time of MPI-P1 and MPI-P2 are significantly reduced. Specifically, for the ship design data, when 20 compute nodes are used, the running time of MPI-P1 and MPI-P2 are 34.4 and 17.3 seconds while the original HMAC takes 1,360.7 seconds to obtain the clustering results. In addition, the performance of these two parallel approaches are very close. MPI-P2 is slightly faster than MPI-P1, possibly due to less data and message exchange.

[1] http://sites.stat.psu.edu/jiali/hmac/doc.pdf

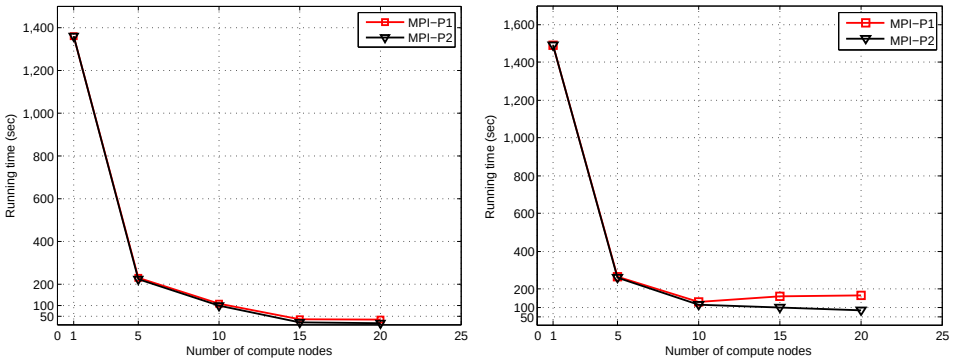


Figure 5.2: Running time of two parallel HMAC approaches using MPI on ship design data (left) and imagery data (right)

5.6 Summary

We propose two parallel versions of HMAC using MPI, an open source parallel computing library. The first parallel approach partitions the computation in each iteration of the inference algorithm into several subtasks, which are conducted independently and concurrently. A master node summarizes all the partial results during each iteration and also groups data into clusters finally. The second approach partitions the data into several subsets, assigns a slave node to compute the modes for the data in each subset using the original MEM, and has a master node do the final merging and clustering. Comparing with the original algorithm, parallel HMAC significantly reduces the running time as the number of compute nodes increases, making it feasible to perform clustering on large-scale data for real applications.

Chapter 6

Conclusions and Future Work

We summary the contributions of this dissertation and discuss some future work.

6.1 Conclusions

This dissertation is focused on mixture modeling for complex and large-scale data and their applications in classification and clustering. A set of new mixture models is proposed to model the distribution of data that have high dimensions, missing values, or are more complicated than those in a vector space, for instance, ones that contain sets of weighted and unordered vectors.

A two-way GMM is proposed to classify high dimensional data and group variables into clusters simultaneously. Variables in the same cluster are assumed to have the same distributions within each class. For each cluster of variables, only a small number of statistics are sufficient for predicting the class label, resulting in a dimension reduction property. We assume a component-wise diagonal covariance matrix, i.e., the variables are independent for each mixture component. This assumption permits the treatment of missing values, a particularly useful trait for data that are prone to missing values. EM algorithms are derived to estimate the two-way GMM with or without missing values. Experimental results show that a two-way mixture often outperforms a mixture model without variable grouping.

A distance-based mixture modeling approach via the concept of hypothetical local mapping (HLM) is proposed to estimate a mixture-type density for the data using their pairwise distances. Since only distances are required for model estima-

tion, HLM is particularly useful for the modeling of data that cannot be effectively described by well-studied mathematical entities. Experimental results show that the classification performance of the proposed mixture model is highly competitive, comparing with other state-of-the-art distance-based classification methods, on various datasets. The computational cost for both training and testing are also low. Because a mixture model is estimated for each class separately, it is easy to handle a large number of classes in the HLM based modeling approach. In addition, it can be extended to the classification of stream data that arrive in an incremental fashion.

The intrinsic characteristics of mixture modeling render it a special tool for visual analytics. Motivated by this, we propose a GMM with the component means constrained in a pre-selected subspace. It is particularly appealing to multidimensional data visualization, in which users may already know that the component (or cluster) means of data lie in a subspace spanned by several dimensions of the data. Therefore, the subspace is fixed and a GMM is estimated with the component means constrained in that subspace. It is proved that the modes, the component means of a Gaussian mixture, and the class means all lie in the same constrained subspace. If the subspace is unknown, this motivates us to find one by applying weighted principal component analysis to the modes, the class means, or a union set of modes and class means. The constrained method has a dimension reduction property, which allows us to view the clustering or classification structure of high dimensional data in a lower dimensional subspace. Although reduced-rank MDA is a competitive method by constraining the component means to an optimal discriminant subspace updated within each iteration of the EM algorithm, experiments on several real datasets show that when the dimension of the discriminant subspace is very low, our proposed method with a simple technique of spanning the subspace using only the class means often outperforms reduced-rank MDA.

We propose parallel implementations of hierarchical mode association clustering (HMAC) using message passing interface (MPI), an open-source parallel computing library. HMAC is a nonparametric clustering method that groups data into one cluster if they are associated with the same mode in a mixture density. Two parallel approaches are tested. The first one partitions the computation in each iteration of the MEM algorithm into subtasks, which are performed independently

and concurrently by several slave nodes. It then has a master node sum up all the partial results and finish the computation for one iteration. The parallelization on each iteration continues until the MEM algorithm converges. The second one partitions the data into several subsets and assigns a slave node to compute the modes of data in each subset using the original MEM algorithm. Since the mode of each individual data can be obtained separately, every slave node can perform the assigned computation independently. Comparing with the original algorithm, parallel HMAC significantly reduces the running time when working on large-scale data.

6.2 Future Work

The works in this dissertation provide the starting point of mixture modeling for complex and large-scale data. We discuss some future work in this section.

For two-way Gaussian mixtures, the variables or features may have physical meanings in engineering systems. Prior knowledge of such physical meanings may be exploited in grouping variables. The two-way mixture approach may be extended to achieve dimension reduction under more general settings.

In the distance-based mixture modeling using hypothetical local mapping (HLM), a common shape parameter is assumed across all the components in all the classes or all the components within a class. This common shape may be limited in some situations since the assumption may be conservative. We may relax it by the quantization trick, in the same spirit as the two-way mixture model. Specifically, we may estimate a shape parameter for every cluster separately, collect all the scale parameters across the components, and then quantize them into a few groups. To improve robustness, we assume common scale parameters for the components that are put in the same group and then estimate common shape parameters for components in the same group. In a nutshell, the scale parameters obtained individually for each component serve as a crude estimation in the first step. Then components with similar scale parameters are pooled together to estimate a common scale parameter for robustness.

In the GMM with the component means constrained in a pre-selected subspace, we select a constrained subspace that has the largest training likelihood among

a sequence of subspaces resulting from different kernel bandwidths. When the number of candidate subspaces is large, it may be desired to narrow down the search by incorporating some prior knowledge. For instance, the proposed method has a potential in visualization when people already know that certain dimensions of the data matter for classification or clustering, i.e., a constrained subspace can be obtained beforehand. This subspace constrained method may be extended to other parametric mixtures, for instance, mixture of Poisson for discrete data.

We propose two parallel approaches to hierarchical mode association clustering (HMAC), a nonparametric method that groups data points into a cluster if they are associated with the same mode in a mixture density. The same approaches can be applied to parallelize ridgeline EM algorithm (Li et al., 2007), which finds the ridgeline linking two hilltops (modes) in a mixture density. The ridgeline can be used to measure how well two hills (cluster of data) separate from each other, enabling the diagnosis of clustering results. The combination of these approaches can help us instantly obtain the geometric characteristics of a mixture density on large-scale data, which provide very useful information for visual analytics.

Appendix A

Two-way Gaussian mixtures

A.1 Dimension Reduction Property

We now prove Theorem 2.3.1. Denote the number of variables in the lth cluster in class k by $\eta_{k,l}$, $\sum_{l=1}^{L}\eta_{k,l}=p$ for all k. Suppose variables in cluster l under class k are $\{j_1^{(k,l)}, j_2^{(k,l)}, ..., j_{\eta_{k,l}}^{(k,l)}\}$. The general two-way mixture model in (2.6) can also be written as

$$\begin{aligned} f(\mathbf{X}=\mathbf{x}, Y=k) &= \sum_{m=1}^{M}\pi_m p_m(k)\prod_{j=1}^{p}\phi\left(x_j|\boldsymbol{\theta}_{m,c(b(m),j)}\right) \\ &= \sum_{m\in\mathcal{R}_k}\pi_m\prod_{l=1}^{L}\prod_{i=1}^{\eta_{k,l}}\phi\left(x_{j_i^{(k,l)}}|\boldsymbol{\theta}_{m,l}\right) . \end{aligned} \tag{A.1}$$

Since the distribution of $x_{j_i^{(k,l)}}$ is from the exponential family, we have

$$\begin{aligned} \prod_{i=1}^{\eta_{k,l}}\phi\left(x_{j_i^{(k,l)}}|\boldsymbol{\theta}_{m,l}\right) &= \prod_{i=1}^{\eta_{k,l}}\exp\left(\sum_{s=1}^{S}\eta_s(\boldsymbol{\theta}_{m,l})T_s(x_{j_i^{(k,l)}})-\mathbf{B}(\boldsymbol{\theta}_{m,l})\right)h\left(x_{j_i^{(k,l)}}\right) \\ &= \exp\left(\sum_{s=1}^{S}\eta_s(\boldsymbol{\theta}_{m,l})\sum_{i=1}^{\eta_{k,l}}T_s(x_{j_i^{(k,l)}})-\eta_{k,l}\mathbf{B}(\boldsymbol{\theta}_{m,l})\right)\prod_{i=1}^{\eta_{k,l}}h\left(x_{j_i^{(k,l)}}\right) \end{aligned} \tag{A.2}$$

We have defined $\overline{\mathbf{T}}_{l,k}(\mathbf{x})=\sum_{i=1}^{\eta_{k,l}}\mathbf{T}(x_{j_i^{(k,l)}})$. More specifically, $\overline{\mathbf{T}}_{l,k}(\mathbf{x})=(\overline{T}_{l,k,1}(\mathbf{x}), ..., \overline{T}_{l,k,S}(\mathbf{x}))^t$, where $\overline{T}_{l,k,s}(\mathbf{x})=\sum_{i=1}^{\eta_{k,l}}T_s(x_{j_i^{(k,l)}})$, $s=1, ..., S$. Substitute (A.2) into (A.1),

$$f(\mathbf{X}=\mathbf{x}, Y=k)$$

$$= \sum_{m\in\mathcal{R}_k} \pi_m \left[\prod_{l=1}^{L} \exp\left(\sum_{s=1}^{S} \eta_s(\boldsymbol{\theta}_{m,l}) \sum_{i=1}^{\eta_{k,l}} T_s(x_{j_i^{(k,l)}}) - \eta_{k,l}\mathbf{B}(\boldsymbol{\theta}_{m,l})\right)\right] \left[\prod_{l=1}^{L}\prod_{i=1}^{\eta_{k,l}} h(x_{j_i^{(k,l)}})\right]$$

$$= \left[\sum_{m\in\mathcal{R}_k} \pi_m \prod_{l=1}^{L} \exp\left(\sum_{s=1}^{S} \eta_s(\boldsymbol{\theta}_{m,l})\overline{T}_{l,k,s}(\mathbf{x}) - \eta_{k,l}\mathbf{B}(\boldsymbol{\theta}_{m,l})\right)\right] \left[\prod_{j=1}^{p} h(x_j)\right] .$$

Because $f(Y = k|\mathbf{X} = \mathbf{x}) \propto f(\mathbf{X} = \mathbf{x}, Y = k)$,

$$f(Y = k|\mathbf{X} = \mathbf{x}) \propto \sum_{m\in\mathcal{R}_k} \pi_m \prod_{l=1}^{L} \exp\left(\sum_{s=1}^{S} \eta_s(\boldsymbol{\theta}_{m,l})\overline{T}_{l,k,s}(\mathbf{x}) - \eta_{k,l}B(\boldsymbol{\theta}_{m,l})\right)$$

subject to $\sum_{k=1}^{K} f(Y = k \mid \mathbf{X} = \mathbf{x}) = 1$. As the posterior probability of Y given $\mathbf{X} = \mathbf{x}$ only depends on $\overline{T}_{l,k,s}(\mathbf{x})$, $\mathbf{X}$ and Y are conditionally independent given $\overline{T}_{l,k,s}(\mathbf{x})$, $l = 1, ..., L$, $k = 1, ..., K$, $s = 1, ..., S$, or equivalently, $\overline{\mathbf{T}}_{l,k}(\mathbf{x})$, $l = 1, ..., L$, $k = 1, ..., K$.

A.2 Model Estimation

The E-step of EM computes $Q(\psi_{t+1}|\psi_t)$ and the M-step maximizes it. $Q(\psi_{t+1}|\psi_t) = E[\log f(\mathbf{v}|\psi_{t+1}) \mid \mathbf{w}, \psi_t]$, where $\mathbf{v}$ is the complete data, $\mathbf{w}$ the incomplete, and $f(\cdot)$ the density function. Let $\tau^{(i)}$ be the latent component identity of $\mathbf{x}^{(i)}$. We abuse the notation $\Lambda(\mathbf{x}^{(i)})$ slightly to mean the non-missing variables in $\mathbf{x}^{(i)}$. Here $\mathbf{v} = \{\mathbf{x}^{(i)}, y^{(i)}, \tau^{(i)} : i = 1, ..., n\}$, and $\mathbf{w} = \{\Lambda(\mathbf{x}^{(i)}), y^{(i)} : i = 1, ..., n\}$. $Q(\psi_{t+1}|\psi_t) = \sum_{i=1}^{n} E\left[\log f(\mathbf{x}^{(i)}, \tau^{(i)}, y^{(i)}|\psi_{t+1}) \mid \Lambda(\mathbf{x}^{(i)}), y^{(i)}, \psi_t\right]$, where

$$\begin{aligned} & E\left[\log f(\mathbf{x}^{(i)}, \tau^{(i)}, y^{(i)}|\psi_{t+1}) \mid \Lambda(\mathbf{x}^{(i)}), y^{(i)}, \psi_t\right] \\ = \; & E\left[\log \pi^{(t+1)}_{\tau^{(i)}} \mid \Lambda(\mathbf{x}^{(i)}), y^{(i)}, \psi_t\right] + E\left[\log p_{\tau^{(i)}}(y^{(i)}) \mid \Lambda(\mathbf{x}^{(i)}), y^{(i)}, \psi_t\right] + \\ & \sum_{j=1}^{p} E\left[\log \phi(x_j^{(i)} \mid \mu^{(t+1)}_{\tau^{(i)},c^{(t+1)}(b(\tau^{(i)}),j)}, {\sigma^2}^{(t+1)}_{\tau^{(i)},c^{(t+1)}(b(\tau^{(i)}),j)}) \mid \Lambda(\mathbf{x}^{(i)}), y^{(i)}, \psi_t\right] \end{aligned} \quad (A.3)$$

Let $q_{i,m}$ be the posterior probability for $\Lambda(\mathbf{x}^{(i)})$ being in component m under ψ_t, as given in Eq.(2.13).

The first term in (A.3), $E[\log \pi^{(t+1)}_{\tau^{(i)}} \mid \Lambda(\mathbf{x}^{(i)}), y^{(i)}, \psi_t] = \sum_{m=1}^{M} q_{i,m} \log \pi_m^{(t+1)}$. The second term in (A.3) is zero. For the third term, consider each j separately. If $x_j^{(i)}$ is not missing, that is, $\Lambda(x_j^{(i)}) = 1$, the distribution of the complete data $\{x_j^{(i)}, \tau^{(i)}, y^{(i)}\}$ conditioned on the incomplete data is random only in terms of

$\tau^{(i)} \in \{1, ..., M\}$, which is the pmf given by the posterior probabilities $q_{i,m}$. Thus,

$$\begin{aligned} & E\left[\log \phi(x_j^{(i)} \mid \mu^{(t+1)}_{\tau^{(i)},c^{(t+1)}(b(\tau^{(i)}),j)}, {\sigma^2}^{(t+1)}_{\tau^{(i)},c^{(t+1)}(b(\tau^{(i)}),j)}) \mid \Lambda(\mathbf{x}^{(i)}), y^{(i)}, \psi_t\right] \\ = & \sum_{m=1}^{M} q_{i,m} \cdot \left[\log \frac{1}{\sqrt{2\pi {\sigma^2}^{(t+1)}_{m,c^{(t+1)}(b(m),j)}}} - \frac{\left(x_j^{(i)} - \mu^{(t+1)}_{m,c^{(t+1)}(b(m),j)}\right)^2}{2{\sigma^2}^{(t+1)}_{m,c^{(t+1)}(b(m),j)}}\right] . \end{aligned}$$

If $x_j^{(i)}$ is missing, that is, $\Lambda(x_j^{(i)}) = 0$, the distribution of the complete data $\{x_j^{(i)}, \tau^{(i)}, y^{(i)}\}$ conditioned on the incomplete data is random in terms of both $\tau^{(i)} \in \{1, ..., M\}$ and the variable $x_j^{(i)}$. The conditional distribution of $\tau^{(i)}$ is still given by the posterior probabilities $q_{i,m}$, $m = 1$, ..., M. The conditional distribution of $x_j^{(i)}$ given $\{\Lambda(\mathbf{x}^{(i)}), y^{(i)}, \tau^{(i)} = m\}$ under ψ_t is $\mathcal{N}(\mu^{(t)}_{m,c^{(t)}(b(m),j)}, {\sigma^2}^{(t)}_{m,c^{(t)}(b(m),j)})$. Thus

$$\begin{aligned} & E[\log \phi(x_j^{(i)} \mid \mu^{(t+1)}_{\tau^{(i)},c^{(t+1)}(b(\tau^{(i)}),j)}, {\sigma^2}^{(t+1)}_{\tau^{(i)},c^{(t+1)}(b(\tau^{(i)}),j)} \mid \Lambda(\mathbf{x}^{(i)}), y^{(i)}, \psi_t] \\ = & \sum_{m=1}^{M} q_{i,m} \cdot \left[\log \frac{1}{\sqrt{2\pi {\sigma^2}^{(t+1)}_{m,c^{(t+1)}(b(m),j)}}} - \frac{\left(\mu^{(t)}_{m,c^{(t)}(b(m),j)} - \mu^{(t+1)}_{m,c^{(t+1)}(b(m),j)}\right)^2 + {\sigma^2}^{(t)}_{m,c^{(t)}(b(m),j)}}{2{\sigma^2}^{(t+1)}_{m,c^{(t+1)}(b(m),j)}}\right] . \end{aligned}$$

In summary, $Q(\psi_{t+1} \mid \psi_t)$ is given by the formula below.
Let

$$\Delta_1 = \frac{(x_j^{(i)} - \mu^{(t+1)}_{m,c^{(t+1)}(b(m),j)})^2}{2{\sigma^2}^{(t+1)}_{m,c^{(t+1)}(b(m),j)}} , \Delta_2 = \frac{\left(\mu^{(t)}_{m,c^{(t)}(b(m),j)} - \mu^{(t+1)}_{m,c^{(t+1)}(b(m),j)}\right)^2 + {\sigma^2}^{(t)}_{m,c^{(t)}(b(m),j)}}{2{\sigma^2}^{(t+1)}_{m,c^{(t+1)}(b(m),j)}} .$$

Then

$$\begin{aligned} & Q(\psi_{t+1} \mid \psi_t) = \sum_{i=1}^{n}\sum_{m=1}^{M} q_{i,m} \log \pi_m^{(t+1)} + \\ & \sum_{i=1}^{n}\sum_{m=1}^{M}\sum_{j=1}^{p} q_{i,m} \cdot \left[\log \frac{1}{\sqrt{2\pi {\sigma^2}^{(t+1)}_{m,c^{(t+1)}(b(m),j)}}} - \left(\Lambda(x_j^{(i)})\Delta_1 + (1 - \Lambda(x_j^{(i)}))\Delta_2\right)\right] \end{aligned}$$

Based on the obtained $Q(\psi_{t+1} \mid \psi_t)$, the formulas for updating the parameters in Eqs.(2.14) $\sim$ (2.17) can be easily derived.

Appendix B

GMM with Subspace Constrained Means

B.1 Proof of Theorem 3.3.1

We prove Theorem 4.3.1. Consider a mixture of Gaussians with a common covariance matrix $\boldsymbol{\Sigma}$ shared across all the components as in (2):

$$f(\boldsymbol{X} = \boldsymbol{x}) = \sum_{r=1}^{R} \pi_r \phi(\boldsymbol{x}|\boldsymbol{\mu}_r, \boldsymbol{\Sigma}) \ .$$

Once $\boldsymbol{\Sigma}$ is identified, a linear transform (a "whitening" operation) can be applied to $\boldsymbol{X}$ so that the transformed data follow a mixture with component-wise diagonal covariance, more specifically, the identity matrix $\boldsymbol{I}$. Assume $\boldsymbol{\Sigma}$ is non-singular and hence positive definite, we can find the natural factor of $\boldsymbol{\Sigma}$, that is, $\boldsymbol{\Sigma} = (\boldsymbol{\Sigma}^{\frac{1}{2}})^t \boldsymbol{\Sigma}^{\frac{1}{2}}$. If the eigen decomposition of $\boldsymbol{\Sigma}$ is $\boldsymbol{\Sigma} = V_{\boldsymbol{\Sigma}} D_{\boldsymbol{\Sigma}} V_{\boldsymbol{\Sigma}}^t$, then, $\boldsymbol{\Sigma}^{\frac{1}{2}} = D_{\boldsymbol{\Sigma}}^{\frac{1}{2}} V_{\boldsymbol{\Sigma}}^t$. Let $W = ((\boldsymbol{\Sigma}^{\frac{1}{2}})^t)^{-1}$ and $\boldsymbol{Z} = W\boldsymbol{X}$. The density of $\boldsymbol{Z}$ is $g(\boldsymbol{Z} = \boldsymbol{z}) = \sum_{r=1}^{R} \pi_r \phi(\boldsymbol{z}|W\boldsymbol{\mu}_r, \boldsymbol{I})$. Any mode of $g(\boldsymbol{z})$ corresponds to a mode of $f(x)$ and vice versa. Hence, without loss of generality, we can assume $\boldsymbol{\Sigma} = \boldsymbol{I}$.

Another linear transform on $\boldsymbol{Z}$ can be performed using the orthonormal basis $V = \boldsymbol{\nu} \cup \boldsymbol{\nu}^{\perp} = \{\boldsymbol{v}_1, ..., \boldsymbol{v}_p\}$, where $\boldsymbol{\nu} = \{\boldsymbol{v}_{q+1}, ..., \boldsymbol{v}_p\}$ is the constrained subspace where $\boldsymbol{\mu}_{kr}$'s reside, and $\boldsymbol{\nu}^{\perp} = \{\boldsymbol{v}_1, ..., \boldsymbol{v}_q\}$ is the corresponding null subspace, as defined in Section 3. Suppose $\widetilde{\boldsymbol{Z}} = \mathbf{Proj}_V^{\boldsymbol{Z}}$. For the transformed data $\widetilde{\boldsymbol{z}}$, the

covariance matrix is still $\boldsymbol{I}$. Again, there is a one-to-one correspondence (via the orthonormal linear transform) between the modes of $g_k(\widetilde{\boldsymbol{z}})$ and the modes of $g_k(\boldsymbol{z})$. The density of $\widetilde{\boldsymbol{z}}$ is

$$g(\widetilde{\boldsymbol{Z}} = \widetilde{\boldsymbol{z}}) = \sum_{r=1}^{R} \pi_r \phi(\widetilde{\boldsymbol{z}}|\boldsymbol{\theta}_r, \boldsymbol{I}) \ ,$$

where $\boldsymbol{\theta}_r$ is the projection of $W\boldsymbol{\mu}_r$ onto the orthonormal basis V, i.e., $\boldsymbol{\theta}_{kr} = \mathbf{Proj}_V^{W\boldsymbol{\mu}_r}$. Split $\boldsymbol{\theta}_r$ into two parts, $\boldsymbol{\theta}_{r,1}$ being the first q dimensions of $\boldsymbol{\theta}_r$ and $\boldsymbol{\theta}_{r,2}$ being the last $p-q$ dimensions. Since the projections of $\boldsymbol{\mu}_r$'s onto the null subspace $\boldsymbol{\nu}^{\perp}$ are the same, $\boldsymbol{\theta}_{r,1}$ are identical for all the components, which is hence denoted by $\boldsymbol{\theta}_{\cdot,1}$. Also denote the first q dimensions of $\widetilde{\boldsymbol{z}}$ by $\widetilde{\boldsymbol{z}}^{(1)}$, and the last $p-q$ dimensions by $\widetilde{\boldsymbol{z}}^{(2)}$. We can write $g(\widetilde{\boldsymbol{z}})$ as

$$g(\widetilde{\boldsymbol{Z}} = \widetilde{\boldsymbol{z}}) = \sum_{r=1}^{R} \pi_r \phi(\widetilde{\boldsymbol{z}}^{(1)}|\boldsymbol{\theta}_{\cdot,1}, \boldsymbol{I}_q)\phi(\widetilde{\boldsymbol{z}}^{(2)}|\boldsymbol{\theta}_{r,2}, \boldsymbol{I}_{p-q}) \ .$$

where $\boldsymbol{I}_q$ indicates a $q \times q$ identity matrix. Since $g(\widetilde{\boldsymbol{z}})$ is a smooth function, its modes have zero first order derivatives. Note

$$\frac{\partial g(\widetilde{\boldsymbol{z}})}{\partial \widetilde{\boldsymbol{z}}^{(1)}} = \frac{\partial \phi(\widetilde{\boldsymbol{z}}^{(1)}|\boldsymbol{\theta}_{\cdot,1}, \boldsymbol{I}_q)}{\partial \widetilde{\boldsymbol{z}}^{(1)}} \sum_{r=1}^{R} \pi_r \phi(\widetilde{\boldsymbol{z}}^{(2)}|\boldsymbol{\theta}_{r,2}, \boldsymbol{I}_{p-q}) \ ,$$

$$\frac{\partial g(\widetilde{\boldsymbol{z}})}{\partial \widetilde{\boldsymbol{z}}^{(2)}} = \phi(\widetilde{\boldsymbol{z}}^{(1)}|\boldsymbol{\theta}_{\cdot,1}, \boldsymbol{I}_q) \sum_{r=1}^{R} \pi_r \frac{\partial \phi(\widetilde{\boldsymbol{z}}^{(2)}|\boldsymbol{\theta}_{r,2}, \boldsymbol{I}_{p-q})}{\partial \widetilde{\boldsymbol{z}}^{(2)}} \ .$$

By setting the first partial derivative to zero and using the fact $\sum_{r=1}^{R} \pi_r \phi(\widetilde{\boldsymbol{z}}^{(2)}|\boldsymbol{\theta}_{r,2}, \boldsymbol{I}_{p-q}) > 0$, we get

$$\frac{\partial \phi(\widetilde{\boldsymbol{z}}^{(1)}|\boldsymbol{\theta}_{\cdot,1}, \boldsymbol{I}_q)}{\partial \widetilde{\boldsymbol{z}}^{(1)}} = 0 \ ,$$

and equivalently

$$\widetilde{\boldsymbol{z}}^{(1)} = \boldsymbol{\theta}_{\cdot,1} \ , \quad \text{the only mode of a Gaussian density.}$$

For any modes of $g(\widetilde{\boldsymbol{z}})$, the first part $\widetilde{\boldsymbol{z}}^{(1)}$ all equal to $\boldsymbol{\theta}_{\cdot,1}$, that is, the projections of the modes onto the null subspace $\boldsymbol{\nu}^{\perp}$ coincide at $\theta_{\cdot,1}$. Hence the modes and

component means lie in the same constrained subspace $\boldsymbol{\nu}$.

B.2 Dimension Reduction Property

We prove Theorem 3.3.2 here. Assume $\boldsymbol{\nu} = \{\boldsymbol{v}_{q+1}, ..., \boldsymbol{v}_p\}$ is the constrained subspace where $\boldsymbol{\mu}_{kr}$'s reside, and $\boldsymbol{\nu}^{\perp} = \{\boldsymbol{v}_1, ..., \boldsymbol{v}_q\}$ is the corresponding null subspace, as defined in Section 3. We use the Bayes classification rule to classify a sample x: $\widehat{y} = \text{argmax}_k\, f(Y = k|\mathbf{X} = \mathbf{x}) = \text{argmax}_k\, f(\mathbf{X} = \mathbf{x}, Y = k)$.

$$f(\boldsymbol{X} = \boldsymbol{x}, Y = k) = a_k f_k(\boldsymbol{x}) \propto a_k \sum_{r=1}^{R_k} \pi_{kr} \exp(-(\boldsymbol{x} - \boldsymbol{\mu}_{kr})^t \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_{kr}))\ . \quad \text{(B.1)}$$

Let $\boldsymbol{V} = \begin{pmatrix} \boldsymbol{v}_1^t \\ \vdots \\ \boldsymbol{v}_p^t \end{pmatrix}$. Matrix $\boldsymbol{V}$ is orthonormal because $\boldsymbol{v}_j$'s are orthonormal by construction. Consider the following cases of $\boldsymbol{\Sigma}$.

$\boldsymbol{\Sigma}$ is an identity matrix

From Eq. (B.1), we have

$$\begin{aligned}
& \sum_{r=1}^{R_k} \pi_{kr} \exp(-(\boldsymbol{x} - \boldsymbol{\mu}_{kr})^t \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_{kr})) \\
= & \sum_{r=1}^{R_k} \pi_{kr} \exp(-(\boldsymbol{x} - \boldsymbol{\mu}_{kr})^t (\boldsymbol{V}^t \boldsymbol{V}) (\boldsymbol{x} - \boldsymbol{\mu}_{kr})) \\
= & \sum_{r=1}^{R_k} \pi_{kr} \exp(-(\boldsymbol{V}\boldsymbol{x} - \boldsymbol{V}\boldsymbol{\mu}_{kr})^t (\boldsymbol{V}\boldsymbol{x} - \boldsymbol{V}\boldsymbol{\mu}_{kr})) \\
= & \sum_{r=1}^{R_k} \pi_{kr} \exp(-\sum_{j=1}^{p} (\breve{x}_j - \breve{\mu}_{kr,j})^2)\ , \qquad \text{(B.2)}
\end{aligned}$$

where $\breve{x}_j = \boldsymbol{v}_j^t \cdot \boldsymbol{x}$, $\breve{\mu}_{kr,j} = \boldsymbol{v}_j^t \cdot \boldsymbol{\mu}_{kr}$, $j = 1, 2, ..., p$. Because $\breve{\mu}_{kr,j} = c_j$, identical across all k and r for $j = 1, \cdots, q$, the first q terms in the sum of exponent in Eq.

(B.2) are all constants. We have

$$\begin{aligned} & \sum_{r=1}^{R_k} \pi_{kr} \exp(-\sum_{j=1}^{p} (\breve{x}_j - \breve{\mu}_{kr,j})^2) \\ \propto & \sum_{r=1}^{R_k} \pi_{kr} \exp(-\sum_{j=q+1}^{p} (\breve{x}_j - \breve{\mu}_{kr,j})^2) \ . \end{aligned}$$

Therefore,

$$f(\boldsymbol{X} = \boldsymbol{x}, Y = k) \propto a_k \sum_{r=1}^{R_k} \pi_{kr} \exp(-\sum_{j=q+1}^{p} (\breve{x}_j - \breve{\mu}_{kr,j})^2) \ .$$

That is, to classify a sample $\boldsymbol{x}$, we only need the projection of $\boldsymbol{x}$ onto the constrained subspace $\boldsymbol{\nu}^{\perp} = \{\boldsymbol{v}_1, ..., \boldsymbol{v}_q\}$.

$\boldsymbol{\Sigma}$ is a non-identity matrix

We can perform a linear transform (a "whitening" operation) on $\boldsymbol{X}$ so that the transformed data have an identity covariance matrix $\boldsymbol{I}$. Find the natural factor of $\boldsymbol{\Sigma}$, that is, $\boldsymbol{\Sigma} = (\boldsymbol{\Sigma}^{\frac{1}{2}})^t \boldsymbol{\Sigma}^{\frac{1}{2}}$. If the eigen decomposition of $\boldsymbol{\Sigma}$ is $\boldsymbol{\Sigma} = V_{\boldsymbol{\Sigma}} D_{\boldsymbol{\Sigma}} V_{\boldsymbol{\Sigma}}^t$, then $\boldsymbol{\Sigma}^{\frac{1}{2}} = D_{\boldsymbol{\Sigma}}^{\frac{1}{2}} V_{\boldsymbol{\Sigma}}^t$. Let $\boldsymbol{Z} = (\boldsymbol{\Sigma}^{-\frac{1}{2}})^t \boldsymbol{X}$. The distribution of $\boldsymbol{Z}$ is

$$g(\boldsymbol{Z} = \boldsymbol{z}, Y = k) = a_k \sum_{r=1}^{R_k} \pi_{kr} \phi(\boldsymbol{z} | \tilde{\boldsymbol{\mu}}_{kr}, \boldsymbol{I}) \ ,$$

where $\tilde{\boldsymbol{\mu}}_{kr} = (\boldsymbol{\Sigma}^{-\frac{1}{2}})^t \boldsymbol{\mu}_{kr}$. According to our assumption, $\boldsymbol{v}_j^t \cdot \boldsymbol{\mu}_{kr} = c_j$, i.e., identical across all k and r for $j = 1, ..., q$. Plugging into $\boldsymbol{\mu}_{kr} = (\boldsymbol{\Sigma}^{\frac{1}{2}})^t \tilde{\boldsymbol{\mu}}_{kr}$, we get $(\boldsymbol{\Sigma}^{\frac{1}{2}} \boldsymbol{v}_j)^t \cdot \tilde{\boldsymbol{\mu}}_{kr} = c_j$, $j = 1, ..., q$. This means for the transformed data, the component means $\tilde{\boldsymbol{\mu}}_{kr}$'s have a null space spanned by $\{\boldsymbol{\Sigma}^{\frac{1}{2}} \boldsymbol{v}_j | j = 1, ..., q\}$. Correspondingly, the constrained subspace is spanned by $\{(\boldsymbol{\Sigma}^{-\frac{1}{2}})^t \boldsymbol{v}_j | j = q+1, ..., p\}$. It is easy to verify that the new null space and constrained subspace are orthogonal, since $(\boldsymbol{\Sigma}^{\frac{1}{2}} \boldsymbol{v}_j)^t \cdot (\boldsymbol{\Sigma}^{-\frac{1}{2}})^t \boldsymbol{v}_{j'} = \boldsymbol{v}_j{}^t \cdot \boldsymbol{v}_{j'} = 0$, $j = 1, ...q$ and $j' = q+1, ..., p$. The spanning vectors for the constrained subspace, $(\boldsymbol{\Sigma}^{-\frac{1}{2}})^t \boldsymbol{v}_j$, $j = q+1, ..., p$, are not orthonormal in general, but there exists an orthonormal basis that spans the same subspace. With

a slight abuse of notation, we use $\{(\mathbf{\Sigma}^{-\frac{1}{2}})^t \boldsymbol{v}_j | j = q+1, ..., p\}$ to denote a $p \times (p-q)$ matrix containing the column vector $(\mathbf{\Sigma}^{-\frac{1}{2}})^t \boldsymbol{v}_j$. For any matrix A of dimension $p \times d$, $d < p$, let the notation $orth(A)$ denote a $p \times d$ matrix whose column vectors are orthonormal and span the same subspace as the column vectors of A. According to B.2, for the transformed data $\boldsymbol{Z}$, we only need the projection of $\boldsymbol{Z}$ onto a subspace spanned by the column vectors of $orth(\{(\mathbf{\Sigma}^{-\frac{1}{2}})^t \boldsymbol{v}_j | j = q+1, ..., p\})$ to compute the class posterior. Note that $\boldsymbol{Z} = (\mathbf{\Sigma}^{-\frac{1}{2}})^t \boldsymbol{X}$. So the subspace that matters for classification for the original data $\boldsymbol{X}$ is spanned by the column vectors of $(\mathbf{\Sigma}^{-\frac{1}{2}}) \times orth(\{(\mathbf{\Sigma}^{-\frac{1}{2}})^t \boldsymbol{v}_j | j = q+1, ..., p\})$. Again, these column vectors are not orthonormal in general, but there exists an orthonormal basis that spans the same subspace. This orthonormal basis is hence spanned by the column vectors of $orth((\mathbf{\Sigma}^{-\frac{1}{2}}) \times orth(\{(\mathbf{\Sigma}^{-\frac{1}{2}})^t \boldsymbol{v}_j | j = q+1, ..., p\}))$. Since $orth((\mathbf{\Sigma}^{-\frac{1}{2}}) \times orth(\{(\mathbf{\Sigma}^{-\frac{1}{2}})^t \boldsymbol{v}_j | j = q+1, ..., p\})) = orth(\{\mathbf{\Sigma}^{-1} \boldsymbol{v}_j | j = q+1, ..., p\})$,[1] the subspace that matters for classification is thus spanned by the column vectors of $orth(\{\mathbf{\Sigma}^{-1} \boldsymbol{v}_j | j = q+1, ..., p\})$.

In summary, only the linear projection of the data onto a subspace with the same dimension as $\boldsymbol{\nu}$ matters for classification.

B.3 Derivation of μ_{kr} in GEM

We derive the optimal $\boldsymbol{\mu}_{kr}$'s under constraint (4) for a given $\mathbf{\Sigma}$. Note that the term in Eq. (3.6) that involves $\boldsymbol{\mu}_{kr}$'s is:

$$-\frac{1}{2} \sum_{k=1}^{K} \sum_{r=1}^{R_k} \sum_{i=1}^{n_k} q_{i,kr} (\boldsymbol{x}_i - \boldsymbol{\mu}_{kr})^t \mathbf{\Sigma}^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_{kr}) \ . \tag{B.3}$$

Denote $\sum_{i=1}^{n_k} q_{i,kr}$ by l_{kr}. Let $\bar{\boldsymbol{x}}_{kr} = \sum_{i=1}^{n_k} q_{i,kr} \boldsymbol{x}_i / l_{kr}$, i.e., the weighted sample mean of the component r in class k. To maximize Eq. (B.3) is equivalent to minimizing the following term (Anderson, 2000):

$$\sum_{k=1}^{K} \sum_{r=1}^{R_k} l_{kr} (\bar{\boldsymbol{x}}_{kr} - \boldsymbol{\mu}_{kr})^t \mathbf{\Sigma}^{-1} (\bar{\boldsymbol{x}}_{kr} - \boldsymbol{\mu}_{kr}) \ . \tag{B.4}$$

[1] Let matrix A be a $p \times p$ square matrix and B be a $p \times d$ matrix, $d < p$. It can be proved that $orth(A \times orth(B)) = orth(A \times B)$.

To solve the above optimization problem under constraint (4.4), we need to find a linear transform such that in the transformed space, the constraint is imposed on individual coordinates (rather than linear combinations of them), and the objective function is a weighted sum of squared Euclidean distances between the transformed $\bar{\boldsymbol{x}}_{kr}$ and $\boldsymbol{\mu}_{kr}$. Once this is achieved, the optimal solution will simply be given by setting those unconstrained coordinates within each component by the component-wise sample mean, and the constrained coordinates by the component-pooled sample mean. We will discuss the detailed solution in the following.

Find the natural factor of $\boldsymbol{\Sigma}$, that is, $\boldsymbol{\Sigma} = (\boldsymbol{\Sigma}^{\frac{1}{2}})^t\boldsymbol{\Sigma}^{\frac{1}{2}}$. If the eigen decomposition of $\boldsymbol{\Sigma}$ is $\boldsymbol{\Sigma} = V_{\boldsymbol{\Sigma}}D_{\boldsymbol{\Sigma}}V_{\boldsymbol{\Sigma}}^t$, then, $\boldsymbol{\Sigma}^{\frac{1}{2}} = D_{\boldsymbol{\Sigma}}^{\frac{1}{2}}V_{\boldsymbol{\Sigma}}^t$. Now perform the following change of variables:

$$\begin{aligned}
& \sum_{k=1}^{K}\sum_{r=1}^{R_k} l_{kr}(\bar{\boldsymbol{x}}_{kr} - \boldsymbol{\mu}_{kr})^t\boldsymbol{\Sigma}^{-1}(\bar{\boldsymbol{x}}_{kr} - \boldsymbol{\mu}_{kr}) \\
= & \sum_{k=1}^{K}\sum_{r=1}^{R_k} l_{kr}\left[\left(\boldsymbol{\Sigma}^{-\frac{1}{2}}\right)^t(\bar{\boldsymbol{x}}_{kr} - \boldsymbol{\mu}_{kr})\right]^t\left[\left(\boldsymbol{\Sigma}^{-\frac{1}{2}}\right)^t(\bar{\boldsymbol{x}}_{kr} - \boldsymbol{\mu}_{kr})\right] \\
= & \sum_{k=1}^{K}\sum_{r=1}^{R_k} l_{kr}(\tilde{\boldsymbol{x}}_{kr} - \tilde{\boldsymbol{\mu}}_{kr})^t(\tilde{\boldsymbol{x}}_{kr} - \tilde{\boldsymbol{\mu}}_{kr}) \ , \qquad \text{(B.5)}
\end{aligned}$$

where $\tilde{\boldsymbol{\mu}}_{kr} = \left(\boldsymbol{\Sigma}^{-\frac{1}{2}}\right)^t\cdot\boldsymbol{\mu}_{kr}$, and $\tilde{\boldsymbol{x}}_{kr} = \left(\boldsymbol{\Sigma}^{-\frac{1}{2}}\right)^t\cdot\bar{\boldsymbol{x}}_{kr}$. Correspondingly, the constraint in (4.4) becomes

$$\left(\boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{v}_j\right)^t\tilde{\boldsymbol{\mu}}_{kr} = \text{constant over } r \text{ and } k \ , \quad j = 1, ..., q \ . \qquad \text{(B.6)}$$

Let $\boldsymbol{b}_j = \boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{v}_j$ and $\boldsymbol{B} = (\boldsymbol{b}_1, \boldsymbol{b}_2, ..., \boldsymbol{b}_q)$. Note that the rank of $\boldsymbol{V} = (\boldsymbol{v}_1, ..., \boldsymbol{v}_q)$ is q. Since $\boldsymbol{\Sigma}^{\frac{1}{2}}$ is of full rank, $\boldsymbol{B} = \boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{V}$ also has rank q. The constraint in (B.6) becomes

$$\boldsymbol{B}^t\tilde{\boldsymbol{\mu}}_{kr} = \boldsymbol{B}^t\tilde{\boldsymbol{\mu}}_{k'r'} \ , \text{for any } r, r' = 1, ..., R_k, \text{ and any } k, k' = 1, ..., K \ . \qquad \text{(B.7)}$$

Now perform a singular value decomposition (SVD) on $\boldsymbol{B}$, i.e., $\boldsymbol{B} = \boldsymbol{U_B}\boldsymbol{D_B}\boldsymbol{V_B}^t$, where $\boldsymbol{V_B}$ is a $q \times q$ orthonormal matrix, $\boldsymbol{D_B}$ is a $q \times q$ diagonal matrix, which is non-singular since the rank of $\boldsymbol{B}$ is q, and U_B is a $p \times q$ orthonormal matrix.

Substituting the SVD of $\boldsymbol{B}$ in (B.7), we get

$$\boldsymbol{V_B D_B U_B^t \tilde{\mu}}_{kr} = \boldsymbol{V_B D_B U_B^t \tilde{\mu}}_{k'r'}, \text{ for any } r,\, r' = 1, ..., R_k, \text{ and any } k,\, k' = 1, ..., K \ ,$$

which is equivalent to

$$\boldsymbol{U_B^t \tilde{\mu}}_{kr} = \boldsymbol{U_B^t \tilde{\mu}}_{k'r'}, \quad \text{for any } r,\, r' = 1, ..., R_k, \text{ and any } k,\, k' = 1, ..., K, \quad \text{(B.8)}$$

because $\boldsymbol{V_B}$ and $\boldsymbol{D_B}$ have full rank. We can augment $\boldsymbol{U_B}$ to a $p \times p$ orthonormal matrix, $\hat{\boldsymbol{U}} = (\boldsymbol{u}_1, ..., \boldsymbol{u}_q, \boldsymbol{u}_{q+1}, ..., \boldsymbol{u}_p)$, where $\boldsymbol{u}_{q+1}$, ..., $\boldsymbol{u}_p$ are augmented orthonormal vectors. Since $\hat{\boldsymbol{U}}$ is orthonormal, the objective function in Eq. (B.5) can be written as

$$\begin{aligned} & \sum_{k=1}^{K} \sum_{r=1}^{R_k} l_{kr} [\hat{\boldsymbol{U}}^t (\tilde{\boldsymbol{x}}_{kr} - \tilde{\boldsymbol{\mu}}_j)]^t \cdot [\hat{\boldsymbol{U}}^t (\tilde{\boldsymbol{x}}_{kr} - \tilde{\boldsymbol{\mu}}_{kr})] \\ = & \sum_{k=1}^{K} \sum_{r=1}^{R_k} l_{kr} (\breve{\boldsymbol{x}}_{kr} - \breve{\boldsymbol{\mu}}_{kr})^t (\breve{\boldsymbol{x}}_{kr} - \breve{\boldsymbol{\mu}}_{kr}) \ , \end{aligned} \quad \text{(B.9)}$$

where $\breve{\boldsymbol{x}}_{kr} = \hat{\boldsymbol{U}}^t \tilde{\boldsymbol{x}}_{kr}$ and $\breve{\boldsymbol{\mu}}_{kr} = \hat{\boldsymbol{U}}^t \tilde{\boldsymbol{\mu}}_{kr}$. If we denote $\breve{\boldsymbol{\mu}}_{kr} = (\breve{\mu}_{kr,1}, \breve{\mu}_{kr,2}, ..., \breve{\mu}_{kr,p})^t$, then the constraint in (B.8) simply becomes

$$\breve{\mu}_{kr,j} = \breve{\mu}_{k'r',j} \ , \text{ for any } r,\, r' = 1, ..., R_k, \text{ and any } k,\, k' = 1, ..., K,\, j = 1, ..., q \ .$$

That is, the first q coordinates of $\breve{\boldsymbol{\mu}}$ have to be common over all the k and r. The objective function (B.9) can be separated coordinate wise:

$$\sum_{k=1}^{K} \sum_{r=1}^{R_k} l_{kr} (\breve{\boldsymbol{x}}_j - \breve{\boldsymbol{\mu}}_{kr})^t (\breve{\boldsymbol{x}}_{kr} - \breve{\boldsymbol{\mu}}_{kr}) = \sum_{j=1}^{p} \sum_{k=1}^{K} \sum_{r=1}^{R_k} l_{kr} (\breve{x}_{kr,j} - \breve{\mu}_{kr,j})^2 \ .$$

For the first q coordinates, the optimal $\breve{\mu}_{kr,j}$, $j = 1, ..., q$, is solved by

$$\breve{\mu}^*_{kr,j} = \frac{\sum_{k'=1}^{K} \sum_{r'=1}^{R_{k'}} l_{k'r'} \breve{x}_{k'r',j}}{\sum_{k'=1}^{K} \sum_{r'=1}^{R_{k'}} l_{k'r'}} = \frac{\sum_{k'=1}^{K} \sum_{r'=1}^{R_{k'}} l_{k'r'} \breve{x}_{k'r',j}}{n} \ , \quad \text{identical over } r \text{ and } k \ .$$

For the remaining coordinates, $\breve{\mu}_{kr,j}$, $j = q+1, ..., p$:

$$\breve{\mu}^*_{kr,j} = \breve{x}_{kr,j} \ .$$

After $\breve{\boldsymbol{\mu}}^*_{kr}$ is calculated, we finally get $\boldsymbol{\mu}_{kr}$'s under the constraint(4.4):

$$\boldsymbol{\mu}_{kr} = (\boldsymbol{\Sigma}^{\frac{1}{2}})^t \hat{\boldsymbol{U}} \breve{\boldsymbol{\mu}}^*_{kr} \ .$$

B.4 Reduced Rank Mixture Discriminant Analysis

The rank restriction can be incorporated into the mixture discriminant analysis (MDA). It is known that the rank-L LDA fit is equivalent to a Gaussian maximum likelihood solution, where the means of Gaussians lie in a L-dimension subspace (Hastie and Tibshirani, 1996). Similarly, in MDA, the log-likelihood can be maximized with the restriction that all the $R = \sum_{k=1}^{K} R_k$ centroids are confined to a rank-L subspace, i.e., rank $\{\mu_{kr}\} = L$.

The EM algorithm is used to estimate the parameters of the reduced rank MDA, and the M-step is a weighted version of LDA, with R "classes". The component posterior probabilities $q_{i,kr}$'s in the E-step are calculated in the same way as in Eq. (4.5), which are conditional on the current (reduced rank) version of component means and common covariance matrix. In the M-step, π_{kr}'s are still maximized using Eq. (4.7). The maximizations of $\boldsymbol{\mu}_{kr}$ and $\boldsymbol{\Sigma}$ can be viewed as weighted mean and pooled covariance maximum likelihood estimates in a weighted and augmented R-class problem. Specifically, we augment the data by replicating the n_k observations in class k R_k times, with the lth such replication having the observation weight $q_{i,kl}$. This is done for each of the K classes, resulting in an augmented and weighted training set of $\sum_{k=1}^{K} n_k R_k$ observations. Note that the sum of all the weights is n. We now impose the rank restriction. For all the sample points $\boldsymbol{x}_i$'s within class k, the weighted component mean is

$$\boldsymbol{\mu}_{kr} = \frac{\sum_{i=1}^{n_k} q_{i,kr} \boldsymbol{x}_i}{\sum_{i=1}^{n_k} q_{i,kr}} \ .$$

Let $q'_{kr} = \sum_{i=1}^{n_k} q_{i,kr}$. The overall mean is

$$\boldsymbol{\mu} = \frac{\sum_{k=1}^{K} \sum_{r=1}^{R_k} q'_{kr} \boldsymbol{\mu}_{kr}}{\sum_{k=1}^{K} \sum_{r=1}^{R_k} q'_{kr}} \ .$$

The pooled within-class covariance matrix is

$$W = \frac{\sum_{k=1}^{K}\sum_{r=1}^{R_k}\sum_{i=1}^{n_k} q_{i,kr}(\boldsymbol{x}_i - \boldsymbol{\mu}_{kr})^t(\boldsymbol{x}_i - \boldsymbol{\mu}_{kr})}{\sum_{k=1}^{K}\sum_{r=1}^{R_k} q'_{kr}} .$$

The between-class covariance matrix is

$$B = \frac{\sum_{k=1}^{K}\sum_{r=1}^{R_k} q'_{kr}(\boldsymbol{\mu}_{kr} - \boldsymbol{\mu})^t(\boldsymbol{\mu}_{kr} - \boldsymbol{\mu})}{\sum_{k=1}^{K}\sum_{r=1}^{R_k} q'_{kr}} .$$

Define $B^* = (W^{-\frac{1}{2}})^T B W^{-\frac{1}{2}}$. Now perform an eigen-decomposition on B^*, i.e., $B^* = V^* D_B V^{*T}$, where $V^* = (v_1^*, v_2^*, ..., v_p^*)$. Let V be a matrix consisting of the leading L columns of $W^{-\frac{1}{2}}V^*$. Considering maximizing the Gaussian log-likelihood subject to the constraints rank $\{\boldsymbol{\mu}_{kr}\} = L$, the solutions are

$$\hat{\boldsymbol{\mu}}_{kr} = WVV^T(\boldsymbol{\mu}_{kr} - \boldsymbol{\mu}) + \boldsymbol{\mu} , \tag{B.10}$$

$$\hat{\boldsymbol{\Sigma}} = W + \frac{\sum_{k=1}^{K}\sum_{r=1}^{R_k} q'_{kr}(\boldsymbol{\mu}_{kr} - \hat{\boldsymbol{\mu}}_{kr})^t(\boldsymbol{\mu}_{kr} - \hat{\boldsymbol{\mu}}_{kr})}{\sum_{k=1}^{K}\sum_{r=1}^{R_k} q'_{kr}} . \tag{B.11}$$

As a summary, in the M-step of reduced rank MDA, the parameters, π_{kr}, $\boldsymbol{\mu}_{kr}$ and $\boldsymbol{\Sigma}$, are maximized by Eqs. (4.7), (B.10), and (B.11), respectively.

Note that the discriminant subspace is spanned by the column vectors of $V = W^{-\frac{1}{2}}V^*$, with the lth discriminant variable as $W^{-\frac{1}{2}}v_l^*$. In general, $W^{-\frac{1}{2}}v_l^*$'s are not orthogonal, but we can find an orthonormal basis that spans the same subspace.

Appendix C

Distance-based Mixture Modeling

C.1 Proof of Equation (3.4)

The collection of distances is $\mathbf{u} = (u_1, u_2, ..., u_N)$, and its corresponding weight is $\mathbf{w} = (w_1, w_2, ..., w_N)$, where $\sum_{j=1}^{M} \sum_{i \in \mathcal{C}_j} w_i = 1$. The ML estimator maximizes the follwoing weighted log likelihood:

$$\begin{aligned} L(\mathbf{u}|s, b_1, b_2, ..., b_M) &= \sum_{j=1}^{M} \sum_{i \in \mathcal{C}_j} w_i \log f(u_i) \\ &= \sum_{j=1}^{M} \sum_{i \in \mathcal{C}_j} w_i \left[(s-1) \log u_i - s \log b_j - \frac{u_i}{b_j} - \log \Gamma(s) \right] . \end{aligned} \quad \text{(C.1)}$$

With a fixed s, $L(\mathbf{u}|s, b_1, b_2, ..., b_M)$ can be maximized individually on every b_j:

$$\begin{aligned} & \max\ L(\mathbf{u}|s, b_1, b_2, ..., b_M) \\ = & \sum_{j=1}^{M} \max \sum_{i \in \mathcal{C}_j} w_i \left[(s-1) \log u_i - s \log b_j - \frac{u_i}{b_j} - \log \Gamma(s) \right] . \end{aligned} \quad \text{(C.2)}$$

Since $\sum_{i \in \mathcal{C}_j} w_i \left[(s-1) \log u_i - s \log b_j - \frac{u_i}{b_j} - \log \Gamma(s) \right]$ is a concave function of b_j, we can obtain its maximum by setting the first derivative to zero:

$$\sum_{i \in \mathcal{C}_j} w_i (-\frac{s}{b_j} + \frac{u_i}{b_j^2}) = 0 . \quad \text{(C.3)}$$

Let

$$\bar{u}_j = \frac{\sum_{i \in \mathcal{C}_j} w_i u_i}{\sum_{i \in \mathcal{C}_j} w_i}$$

be the weighted average distance for prototype j. b_j is solved by

$$b_j = \frac{\bar{u}_j}{s} \tag{C.4}$$

Now substitute Equation (C.4) into (C.2):

$$\begin{aligned} & \max L(\mathbf{u}|s) \\ = & \sum_{j=1}^{M} \max \sum_{i \in \mathcal{C}_j} w_i \left[s \log s + s \cdot (\log \frac{u_i}{\bar{u}_j} - \frac{u_i}{\bar{u}_j}) - \log \Gamma(s) - \log u_i \right] . \end{aligned}$$

Since $\log \Gamma(s)$ is a convex function of s, it is easy to show that $L(\mathbf{u}|s)$ is also a convex function of s. The maximum of $L(\mathbf{u}|s)$ is thus determined by setting its first derivative to zero:

$$\sum_{j=1}^{M} \sum_{i \in \mathcal{C}_j} w_i \log s + \sum_{j=1}^{M} \sum_{i \in \mathcal{C}_j} w_i \log \frac{u_i}{u_j} - \sum_{j=1}^{M} \sum_{i \in \mathcal{C}_j} w_i \psi(s) = 0 ,$$

which is equivalent to:

$$\log \widehat{s} - \psi(\widehat{s}) = \log \left[\frac{\prod_{j=1}^{M} \bar{u}_j^{\sum_{i \in \mathcal{C}_j} w_i}}{\prod_{i=1}^{N} u_i^{w_i}} \right] . \tag{C.5}$$

Bibliography

A. A. Alizadeh et al. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature*, 403:503-511, 2000.

T. W. Anderson. *An Introduction to Multivariate Statistical Analysis*. Wiley, 2000.

J. D. Banfield and A. E. Raftery. Model-based Gaussian and non-Gaussian clustering. *Biometrics*, 49:803-821, 1993.

M. Balcan and A. Blum. On a theory of learning with similarity functions. In *Proc. ICML*, pages 73-80, 2006.

D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *Proc. ACM SIGKDD*, pages 407-416, 2000.

S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(4):509-522, 2002.

L. Breiman and R. Ihaka. Nonlinear discriminant analysis via scaling and ACE. *Technical Report 40*, Department of Statistics, University of California, Berkeley, California, 1984.

L. Cazzanti, M. R. Gupta, and A. J. Koppal. Generative models for similarity-based classification. *Pattern Recognition*, 41(7):2289-2297, 2008.

L. Cazzanti and M. R. Gupta. Local similarity discriminant analysis. In *Proc. ICML*, pages 137-144, 2007.

L. Cazzanti. Generative models for similarity-based classification. *Ph.D. Thesis*, Department of Electrical Engineering, University of Washington, 2007.

G. Celeux and G. Govaert. A classification EM algorithm for clustering and two stochastic versions. *Computational Statistics and Data Analysis*, 14:315-332, 1992.

C.C. Chang and C.J. Lin. (2001). LIBSVM : a library for support vector machines. [Online]. Available: http://www.csie.ntu.edu.tw/cjlin/libsvm.

Y. Chen, E. K. Garcia, M. R. Gupta, A. Rahimi, and L. Cazzanti. Similarity-based classification: concepts and algorithms. *Journal of Machine Learning Research*, 10:747-776, 2009.

W. Y. Chen, Y. Song, H. Bai, C. J. Lin, and E. Y. Chang. Parallel spectral clustering in distributed systems. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 33(3):568-586, 2011.

G. Cornuejols, M. L. Fisher, and G. L. Nemhauser. Location of bank accounts to optimize float: an Analytic Study of Exact and Approximate Algorithms. *Management Science*, 23:789-810, 1977.

S. Cost and S. Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10(1):57-78, 1993.

J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Comm. ACM*, 51:107-113, 2008.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society Series B*, 39(1):1-38, 1977.

R. P.W. Duin, E. Pekalska, and D. de Ridder. Relational discriminant analysis. *Pattern Recognition Lett.*, 20:1175-1181, 1999.

M. Evans, N. Hastings, and B. Peacock. *Statistical Distributions, 3rd ed.*, John Wiley & Sons, Inc., 2000.

M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc. of the National Academy of Science*, 95:14863-14868, 1998.

R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179-188, 1936.

J. H. Friedman. Regularized discriminant analysis. *Journal of the American Statistical Association*, 84(405):165-175, 1989.

C. Fraley and A. E. Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, 97:611-631, 2002.

T. Graepel, R. Herbrich, and K. Obermayer. Classification on pairwise proximity data. *Adv. Neural Inform. Process. Syst.*, 11:438-444, 1999.

A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman. From few to many: illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(6):643-660, 2001.

Z. Ghahramani and G. E. Hinton. The EM algorithm for factor analyzers. *Technical Report CRG-TR-96-1*, The University of Toronto, Toronto, 1997.

L. Goldfarb. A new approach to pattern recognition. *Prog. Pattern Recognition* 2:241-402, 1985.

W. Gropp, E. Lusk, and A. Skjellum. Using MPI: portable parallel programming with the message-passing interface. *Scientific and engineering computation*, MIT Press, 1999.

Y. Guo, T. Hastie, and R. Tibshirani. Regularized linear discriminant analysis and its application in microarrays. *Biostatistics*, 8(1):86-100, 2006.

M. R. Gupta, R. M. Gray, and R. A. Olshen. Nonparametric supervised learning by linear interpolation with maximum entropy. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(5):766-781, 2006.

M. Hall et al. The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11(1):10-18, 2009.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, 2001.

T. Hastie and R. Tibshirani. Discriminant analysis by Gaussian mixtures. *Journal of the Royal Statistical Society Series B*, 58:155-176, 1996.

X. He, S. Yan, Y. Hu, P. Niyogi, and H. Zhang. Face recognition using laplacianfaces. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(3):328-340, 2005.

S. Hochreiter and K. Obermayer. Support vector machines for dyadic data. *Neural Computation*, 18(6):1472-1510, 2006.

C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines. *IEEE Trans. Neural Networks*, 13(2):415-425, 2002.

Y. Huang, K. B. Englehart, B. Hudgins, and A. D. C. Chan. A Gaussian mixture model based classification scheme for myoelectric control of powered upper limb prostheses. *IEEE Trans. Biomedical Engineering*, 52:1801-1811, 2005.

D. W. Jacobs, D. Weinshall, and Y. Gdalyahu. Classification with nonmetric distances: image retrieval and class representation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(6):583-600, 2000.

A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264-323, 1999.

P. Kenny, P. Ouellet, N. Dehak, and V. Gupta. A study of interspeaker variability in speaker verification. *IEEE Trans. Audio, Speech and Language Processing*, 16(5):980-987, 2008.

R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273-324, 1997.

L. Lazzeroni and A. Owen. Plaid models for gene expression data. *Statistica Sinica*, 12(1):61-86, 2002.

W. Lam, C. Keung, and D. Liu. Discovering useful concept prototypes for classification based on filtering and abstraction. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(8):1075-1090, 2002.

K. Lang. NewsWeeder: Learning to Filter Netnews. In *Proc. ICML*, pages 331-339, 1995.

K. C. Lee, J. Ho, and D. Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(5):684-698, 2005.

J. Li, J. Z. Wang, and G. Wiederhold. IRM: integrated region matching for image retrieval. In *Proc. ACM MM*, pages 147-156, 2000.

J. Li and J. Z. Wang. Real-time computerized annotation of pictures. In *Proc. ACM MM*, pages 911-920, 2006.

J. Li, S. Ray, and B. G. Lindsay. A nonparametric statistical approach to clustering via mode identification. *Journal of Machine Learning Research*, 8(8):1687-1723, 2007.

J. Li and J. Z. Wang. Real-time computerized annotation of pictures. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 30(6):985-1002, 2008.

J. Li and H. Zha. Two-way Poisson mixture models for simultaneous document classification and word clustering. *Computational Statistics and Data Analysis*, 50:163-180, 2006.

L. Lovász and M. D. Plummer. *Matching Theory*. Akadémiai Kiadó - North Holland, Budapest, 1986.

M. Lozano, J. M. Sotoca, J. S. Sanchez, F. Pla, E. Pekalska, and R. P.W. Duin. Experimental study on prototype optimisation algorithms for prototype-based classification in vector spaces. *Pattern Recognition* 39:1827-1838, 2006.

C. L. Mallows, A note on asymptotic joint normality. *Annals of Mathematical Statistics*, 43(2):508-515, 1972.

F. E. Maranzana. On the location of supply points to minimize transport costs. *Opnal. Res. Quart*, 15:261-270, 1964.

G. J. McLachlan and D. Peel. *Finite Mixture Models*. Wiley, 2000.

G. J. McLachlan, D. Peel, and R. W. Bean. Modeling high-dimensional data by mixtures of factor analyzers. *Computational Statistics and Data Analysis* 41:379-388, 2003

G. J. McLachlan and D. Peel. Mixtures of factor analyzers. In *Proc. ICML*, pages 599-606, 2000.

A. McCallum. (1996). Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. [Online]. Available: http://www.cs.cmu.edu/mccallum/bow

G. J. McLachlan and D. Peel. *Finite Mixture Models*, New York: Wiley, 2000.

J. M. Mulvey and H. P. Crowder. Cluster analysis: an application of lagrangian relaxation. *Management Science*, 25:329-340, 1979.

M. Ouyang et al. Gaussian mixture clustering and imputation of microarray data. *Bioinformatics*, 20:917-923, 2004.

P. Pacheco. *An Introduction to Parallel Programming.* Morgan Kaufmann Publishers Inc., 2011.

W. Pan and X. Shen. Penalized model-based clustering with application to variable selection. *Journal of Machine Learning Research*, 8:1145-1164, 2007.

E. Pekalska, P. Paclic, and R. P.W. Duin. A generalized kernel approach to dissimilarity-based classification. *Journal of Machine Learning Research* 2:175-211, 2001.

E. Pekalska, R. P.W. Duin, and P. Paclik. Prototype selection for dissimilarity based classifiers. *Pattern Recognition Lett.*, 39:189-208, 2006.

D. Povey, et al. The subspace gaussian mixture model - a structured model for speech recognition. *Computer Speech and Language*, 25(2):404-439, 2011.

M. Qiao and J. Li. Two-way Gaussian mixture models for high dimensional classification. *Statistical Analysis and Data Mining*, 3(4):259-271, 2010.

M. Qiao and J. Li. Gaussian mixture Models with component means constrained in pre-selected subspaces. *Journal of Computational and Graphical Statistics (submitted)*, 2012.

M. R. Rao. Cluster analysis and mathematical programming. *Journal of the American Statistical Association*, 6:622-626, 1971.

S. Ray and B. G. Lindsay. The topography of multivariate normal mixtures. Annals of Statistics, 33(5):2042-2065, 2005.

D. A. Reynolds, T. F. Quatieri, and R. B. Dunn. Speaker verification using adapted Gaussian mixture models. *Digital Signal Processing*, 10:19-41, 2000.

G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Comm. ACM*, 18(11):613-620, 1975.

B. Schölkopf, C. J. C. Burges, and A. J. Smola. *Advances in kernel methods: support vector learning*, MIT Press, 1999.

R. N. Shepard. The analysis of proximities: multidimensional scaling with an unknown distance function, I. *Psychometrika*, 27(2):125-140, 1962.

R. N. Shepard. The analysis of proximities: multidimensional scaling with an unknown distance function, II. *Psychometrika*, 27(2):219-246, 1962.

P. Simard, Y. L. Cun, and J. Denker. Efficient pattern recognition using a new transformation distance. *Adv. Neural Inform. Process. Syst.*, 5:50-68, 1993.

Y. Song, Z. Zhuang, H. Li, Q. Zhao, J. Li, W.-C. Lee, and C. L. Giles. Real-time automatic tag recommendation. In *Proc. ACM SIGIR*, pages 515-522, 2008.

M. B. Teitz and P. Bart. Heuristic methods for estimating the generalized vertex median of a weighted graph. *Operations Research*, 16:955-961, 1968.

R. Tibshirani, T. Hastie, B. Narasimhan, and G. Chu. Class prediction by nearest shrunken centroids, with applications to DNA microarrays. *Statistical Science*, 18(1):104-117, 2003.

R. Tibshirani and G. Walther. Cluster validation by prediction strength. *Journal of Computational and Graphical Statistics*, 14(3):511-528, 2005.

L. Wang, C. Yang, and J. Feng. On learning with dissimilarity functions. In *Proc. ICML*, pages 991-998, 2007.

S. Wang and J. Zhu. Variable selection for model-based high-dimensional clustering and its application to microarray Data. *Biometrics*, 64:440-448, 2008.

J. H. Ward. Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association*, 58(301): 236-244, 1963.

D. Weinshall, D.W. Jacobs, and Y. Gdalyahu. Classification in non-metric spaces. *Adv. Neural Inform. Process. Syst.*, 11:838-844, 1999.

X. Yan, M. Qiao, J. Li, T. W. Simpson, G. M. Stump, and X. L. Zhang. A work-centered visual analytics model to support engineering design with interactive visualization and data-Mining. In *Proc. IEEE HICSS*, pages 1845-1854, 2012.

X. Yan, M. Qiao, Y. Zhao, T. W. Simpson, G. M. Stump, J. Li, and X. L. Zhang. Work-centered visual analytics: an approach to bridge information visualization and data mining. *ACM Trans. Interactive Intelligent Systems (submitted)*, 2012.

L. Yao, P. Suryanarayan, M. Qiao, J. Z. Wang, and J. Li. OSCAR: on-site composition and aesthetics feedback through exemplars for photographers. *International Journal of Computer Vision*, 96(3):353-383, 2012.

G. Young and A. S. Householder. Discussion of a set of points in terms of their mutual distances. *Psychometrika*, 3:19-22, 1938.

H. Zha, X. He., C. Ding, M. Gu, and H. Simon. Bipartite graph partitioning and data clustering. In *Proc. ACM CIKM*, pages 25-32, 2001.

H. Zhang, A. C. Berg , M. Maire, and J. Malik. SVM-KNN: discriminative nearest neighbor classification for visual category recognition. In *Proc. IEEE CVPR*, pages 2126-2136, 2006.

Vita

Mu Qiao

Mu Qiao entered the Ph.D. program in Computer Science and Engineering at Penn State University in August 2007. He also entered the concurrent M.S. program in Statistics at Penn State in September 2009. Prior to his Ph.D. study, he received the B.E. degree in Computer Science and Engineering from Harbin Institute of Technology in July 2007. He was an undergraduate exchange student in Computer Science and Engineering at Hong Kong University of Science and Technology in 2006. He worked as summer intern at eBay Research Labs, IBM T. J. Watson Research Center, Palo Alto Research Center (PARC), and IBM Almaden Research Center, in 2009, 2010, 2011, and 2012 respectively. He is a recipient of the 2010 American Statistical Association (ASA) Statistical Learning and Data Mining Section Student Paper Competition Award. His research interests include statistical machine learning, data mining, image retrieval, visual analytics, and social computing.