TOWARDS FINDING OPTIMAL MIXTURE OF SUBSPACES FOR DATA CLASSIFICATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

MOHAMED ELHAFIZ MUSTAFA MUSA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

THE DEPARTMENT OF COMPUTER ENGINEERING

OCTOBER 2003

Approval of the Graduate School of Natural and Applied Sciences.

Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of DOCTOR OF PHILOSOPHY.

Prof. Dr. Ayşe Kiper
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of DOCTOR OF PHILOSOPHY.

Assoc.Prof. Dr. Volkan Atalay
Supervisor

Examining Committee Members

Prof. Dr. Neşe Yalabik _____

Prof. Dr. Uğur Halici _____

Prof. Dr. Robert Duin _____

Assoc. Prof. Dr. Volkan Atalay _____

Dr. Dick de Ridder _____

ABSTRACT

TOWARDS FINDING OPTIMAL MIXTURE OF SUBSPACES FOR DATA CLASSIFICATION

MOHAMED ELHAFIZ MUSTAFA MUSA

Ph.D., Department of Computer Engineering

Supervisor: Assoc.Prof. Dr. Volkan Atalay

October 2003, 85 pages

In pattern recognition, when data has different structures in different parts of the input space, fitting one global model can be slow and inaccurate. Learning methods can quickly learn the structure of the data in local regions, consequently, offering faster and more accurate model fitting. Breaking training data set into smaller subsets may lead to curse of dimensionality problem, as a training sample subset may not be enough for estimating the required set of parameters for the submodels. Increasing the size of training data may not be at hand in many situations. Interestingly, the data in local regions becomes more correlated. Therefore, by decorrelation methods we can reduce data dimensions and hence the number of parameters. In other words, we can find uncorrelated low dimensional subspaces that capture most of the data variability. The current subspace modelling methods have proved better performance than the global modelling methods for the given type of training data structure. Nevertheless these methods still need more research work as they are suffering from two limitations

- There is no standard method to specify the optimal number of subspaces.
- There is no standard method to specify the optimal dimensionality for each

subspace.

In the current models these two parameters are determined beforehand. In this dissertation we propose and test algorithms that try to find a suboptimal number of principal subspaces and a suboptimal dimensionality for each principal subspaces automatically.

Keywords: Principal Component Analysis, Mixture Model, Expectation Maximization algorithm.

ÖZ

VERİ SINIFLAMA İÇİN EN İYİ ALTUZAY KARIŞIMLARININ BULUNMASINA DOĞRU

MOHAMED ELHAFIZ MUSTAFA MUSA
Doktora, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Volkan Atalay

Ekim 2003, 85 sayfa

Örüntü tanımada, verilerin girdi uzayında değişik yapıları olduğunda, bu veriye uygun tek bir genel model bulmak çok yavaş ve yanlış olabilir. Öte yandan öğrenme algoritmaları yerel bölgelerdeki verinin yapısını çabucak öğrenebilir ve böylece daha hızlı ve doğru model bulmaya yolaçabilirler. Öğrenme veri kümesini küçük altkümelere bölmek, altkümedeki verinin büyüklüğü nedeniyle altmodelin parameter kümesini bulmak için yeterli olmayabilir. Birçok durumda da eldeki veri kümesi büyütülemeyebilir. Yerel bölgelerdeki veriler ilginçtir ki birbirleriyle daha ilişkili olabilirler. Bu yüzden, dekorelasyon yöntemleriyle verinin boyutunu ve böylece de parameter sayısını indirgeyebiliriz Başka bir deyişle, verinin değişiminin çoğunu yakalayan kendi içinde korele olmayan alçak boyutlu altuzaylar bulabiliriz. Şu anda varolan altuzay modelleme yöntemleri verilen bir öğrenme veri yapısı tipi için genel modelleme yöntemlerine gore daha yüksek başarı göstermişlerdir. Bunun yanısıra, aşağıdaki sınırlamalardan dolayı bu konuda araştırma yapılması gerekmektedir

- Optimal altuzay sayısını bulmak için standart bir yöntem mevcut değildir.
- Herbir altuzaydaki optimal boyutu bulmak için standart bir yöntem mevcut

değildir.

Şu andaki modellerde yukarıda bahsedilen her iki parametre de elle önceden saptanmaktadır. Bu araştırmada suboptimal altuzay sayısını ve herbir altuzay için suboptimal boyutu otomatik bulmak için algoritmaların tasarlanması ve denenmesi önerilmektedir.

Anahtar Kelimeler: Temel bileşen analizi, Karışım modeli, Bekleme-eniyileme algorıtması.

To my mother and the soul of my father

ACKNOWLEDGMENTS

This thesis could no have been produced without the support and assistance of numerous people, for whom I would like to include this acknowledgement.

Most of all, I wish to thank my supervisor, Assoc. Prof. Volkan Atalay, for his excellent guidance, his patience and extremely friendly and supportive attitude. I would like also to thank all the colleagues in Computer engineering department in METU.

Secondly I would like to express my gratitude to the great scientist Prof. Robert Duin and Dr. Dick De Ridder who introduced me to the key concepts of the problems involved in this thesis. They have given me valuable help and guidance when I was with then in TUDelft and remotely after I came back to Ankara. I would like also to thank all the colleagues in the pattern recognition group in TUDelft, especially the neural network subgroup.

I am also grateful to my jury and follow-up committee members, our grand supervisor Prof. Neşe Yalabık and Prof. Uğur Halıcı.

Thanks are also due to Prof. Turhan Alper for his encouragements and support and the colleagues in Computer engineering department in Çankaya university.

Finally I would like to thank my wife Nagat Hassan for her patience and understanding.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 Overview

In pattern recognition and related fields, each object is represented by a vector of d features or measurements and it is viewed as a point in a d-dimensional space. For instance, if we are dealing with a digitized $n \times n$ image, we convert it into $(n \times n)$-dimensional vector. Treating this high dimensional vector without any reduction in its dimension is computationally infeasible. This is one of the reasons behind developing the so called *feature extraction methods*. Principal component analysis (henceforth PCA) is the most popular feature extraction method, that exploits data correlation, for compression, visualization and classification (Jolliffe 2002).

PCA was introduced early in the past century (Pearson 1901). Nevertheless, it still undergoes research and new extensions are found. Moreover, recently with the emergence of neural networks, PCA has been cast as a neural network feature extraction approach (Haykin 1999). PCA finds a subspace, called *principal subspace*, inside the data space in which the data is uncorrelated. This objective together with the property of being the optimal reconstruction error linear transform, make PCA one of the most popular techniques in many fields like pattern recognition, machine learning, data mining and data compression.

When the underlying data have different structures in different regions of the input space or the data is distributed near to a curve or a nonlinear manifold, a low

dimensional principal subspace can poorly fit it. Figure 1.1 illustrates this problem and its possible solutions using a data set which has a banana shaped distribution. It is clear form the figure that the principal subspace found by conventional PCA method does not fit the given data set properly. This problem is a direct consequence of PCA global linearity. Currently, there are two methods for dealing with this problem: [1] by using one of the PCA nonlinear extensions—most of these extensions are neural based— (Bailing et al. 2001, DeMers & Cottrell 1993, Petsche et al. 1996, Stanford & Raftery 1997). [2] by modelling the data nonlinearity and heterogeneity by a mixture of local linear principal subspaces model (Kambhatla 1995, Moghaddam & Pentland 1997, Sung & Poggio 1998, Tipping & Bishop 1999*a*). Figure 1.1 illustrates these methods graphically. Furthermore, the result of many recent studies suggest that mixtures of local subspaces can be more accurate and faster than a global nonlinear methods (Kambhatla & Leen 1997, Moghaddam 2002).

1.2 Problem definition

A number of implementations of "mixtures of local PCA" (henceforth MPCA) have been proposed in the literature, each defining a different algorithm or a variation. Dony & Haykin (1997), Kambhatla & Leen (1997) all proposed a kind of Vector Quantization (VQ)/PCA mixture models, which first partition the data into disjoint regions by (VQ) and then perform a local PCA about each region center. In these studies, the reconstruction error is employed as the relevant distortion measure for determining the partition. A turning study in the history of this model came from Hinton et al. (1997), as their work is the first work that used one global expectation maximization (EM) training process in a pseudo-likelihood framework. All algorithms proposed before this one have two separate processes; partitioning the data space by *hard clustering* methods, followed by fitting a PCA for each cluster. Another major enhancement came from Moghaddam & Pentland (1997), Roweis (1997), Tipping & Bishop (1999*a*) through proposing the *Probabilistic Principal Component Analyzer* (henceforth PPCA). By giving a probabilistic definition for PCA, the usage of the mixture model and *soft clustering* to define the mixture of PPCA (henceforth MPPCA)

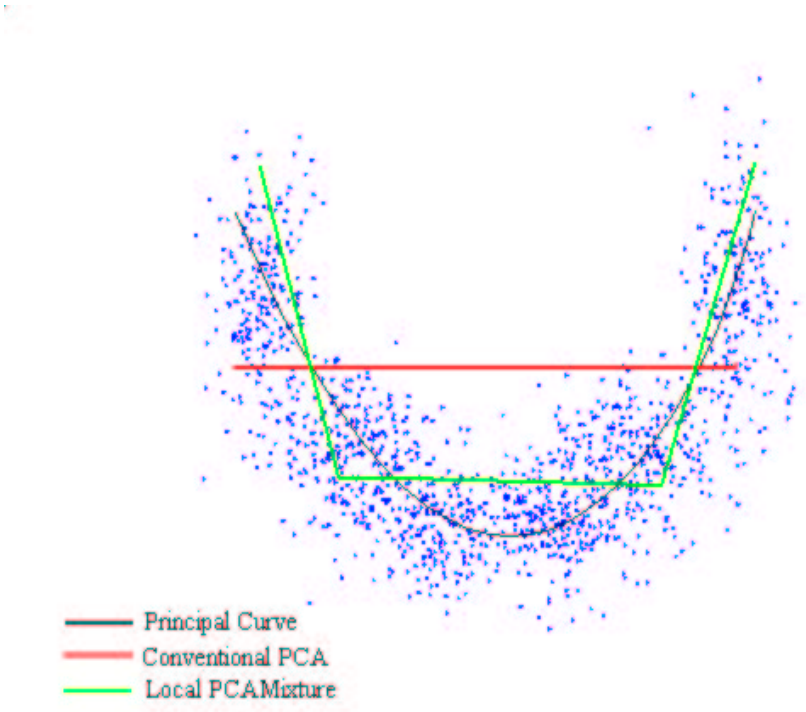


Figure 1.1: The red line shows the conventional PCA global linear 1-dimensional subspace. The black curve represents the best fit difficult to find principal curve. The green lines represent the state-of-the-art nonlinearity approximation by a set of local linear principal subspaces

is straight forward.

All of the above models, while they provide many enhancements, they still have many problems that need more studies. These problems can be summarized in the following points

- The number of subspaces k must be specified beforehand.
- The dimensionality m for subspaces must be specified beforehand.
- EM Problems: EM is the standard method for mixture training. EM is highly dependent on initialization. For this reason any EM-based learning scheme should cater for this problem.

The main aim of this thesis is defining a complete greedy scheme for mixture of PCA training that alleviates the problems mentioned above.

1.3 Contribution and organization

The main contributions of this thesis can be stated as follows

- Defining an EM-based greedy learning algorithm for the MPPCA model, that finds suboptimal number of principal subspaces k automatically.
- Defining a greedy hard clustering initialization scheme, which can be incorporated seamlessly with the above EM-base scheme to alleviate EM initialization problem and to speed up the model training process.
- Equip the scheme with a retained variance method for finding suitable dimensionality m for each subspace, which allows all subspaces to retain the same local variability, by keeping different local dimensionalities i.e., different value for m in each subspace.
- Test the scheme on two classification problems: Handwritten digit classification problem and Ionosphere signal classification problem.
- Test the scheme on texture image segmentation problem.

The thesis is organized as follows. Chapter 2 is devoted for explaining principal subspace. This chapter starts by explaining the conventional way of learning the principal subspace. The chapter then explains the major drawbacks of the conventional PCA and the variant suggestions to overcome these problems. Chapter 2 gives considerable part for explaining the probabilistic PCA, which enables a formal definition for the PCA mixture model. Chapter 3 discusses the mixture of principal subspaces. The chapter also explains the hard clustering methods and soft clustering methods (mixture models). The chapter then gives detailed description for the MPPCA greedy learning scheme's algorithms. Chapter 4 and Chapter 5 present the experimental results of the model. As usual we devote Chapter 6 for conclusions and future work.

1.4 Notations and abbreviations

In general, throughout this thesis, scalar variable is denoted by lowercase italic letters, e.g., i, while matrices and vectors are denoted by boldface letters, e.g., $\mathbf{A}$. Furthermore, matrices are denoted by uppercase, e.g. $\mathbf{C}$ and vectors are denoted by lowercase, e.g., $\mathbf{y}$ Table 1.1 below, list the notation and abbreviation adopted in this thesis.

Table 1.1: Notation and Abbreviation

Symbol	Meaning
FA	Factor Analysis
FD-MPPCA	Fixed Dimensionality Mixture of Probabilistic Principal Component Analyzers
EM	Expectation Maximization
MoG	Mixture of Gaussians
MPPCA	Mixture of Probabilistic Principal Component analyzers
PC	Principal Component
PCA	Principal Component Analysis
PPCA	Probabilistic Principal Component Analysis
VD-MPPCA	Variable Dimensionality Mixture of Probabilistic Principal Component Analyzers
$\mathbf{A}$	PCA projection matrix
$\mathbf{C}$	Model covariance matrix
D	Training data set
d	Dimension in data space
f_k	Mixture density function of k compnents
$\mathbf{I}$	Identity matrix
k	Number of components in a mixture
m	Dimension in principal subspace
N	Number of data element in a training set
p	Density function
P	Probability function
$\mathbf{S}$	Data covariance matrix
$\mathbf{x}$	Vector in subspace
$\mathbf{y}$	Observed data vector
λ	Eigenvalue
α	Retained variance ratio
α_{opt}	Suboptimal retained variance ratio
$\boldsymbol{\theta}$	Parameter vector
$\boldsymbol{\mu}$	Mean
σ^2	Noise in the PPCA model

CHAPTER 2

Principal subspaces

2.1 Curse of dimensionality

A simple table-lookup technique for classification (partitioning the feature space into cells and associate each cell with a class label) requires the number of training data points to be an exponential function of the feature dimension (Bishop 1995). Bellman (1961) termed this phenomenon as "curse of dimensionality". Of course, the technique of dividing up the input space into cells and labelling them is inefficient. However, even other complex and more efficient classifiers suffer from this problem. For instance, consider the nearest mean classifier, which entails calculating means of all classes and classifying a data point by measuring its distances from these means and assigning it to the nearest mean class. If we need h data points to estimate the mean of a 1-dimensional class, then we need h^n data points to estimate the mean of n-dimensional class. In other words, a plausible data size for an n-dimensional class mean estimation is around $\mathcal{O}(2^n)$. This is an astronomical figure for high dimensional data and no practical training data set is expected to fulfil this requirement. It is well known that the probability of misclassification of a decision rule does not increase as the number of features increases, as long as the class-conditional densities are completely known (or, equivalently the number of training samples is arbitrary large and therefore representatives of the underlying densities). However, it has been often observed in practice that the added features may actually degrade the performance of a classifier

if the number of training data points used to design the classifier is small relative to the number of features (Jain et al. 2000). This paradoxical behavior is referred to as the peaking phenomenon (Raudys & Pikelis 1980*b*, Raudys & Jain 1991). A simple explanation for this phenomenon is as follows: the most commonly used classifiers estimate the unknown parameters and use them in the place of the true parameters in the class-conditional densities. For a fixed sample size, as the number of features is increased (with a corresponding increase in the number of unknown parameters), the reliability of the parameter estimates decrease. Consequently, the performance of the resulting classifier, for a fixed sample size, may degrade with an increase in the number of features. In such cases the classifier designer has two choices: increase the sample size, which is impractical in most situations. The other alternative which is more practical and common, is to reduce the dimensionality of the data, to arrive at a dimensionality that (hopefully) gives the optimal result in accuracy and efficiency.

2.2 How to decrease the dimensionality?

A simple answer for this question, is by searching for features, which when removed from the feature set the classification error does not increase or more interestingly, the classification error decreases. Such methods, are called feature selection. In this concern, it is important to make distinction between *feature selection* and *feature extraction*. The term feature selection refers to algorithms that select the (hopefully) most informative subset of larger set of original features. Methods that create new features based on transformations or combinations of the original feature set are called feature extraction methods. The choice between feature selection and feature extraction depends on the application domain and the specific training data at hand.

2.2.1 Feature selection

Although there are many algorithms for feature selection, we should make it clear: If the goal is to find the optimal feature set, there is no substitute for trying them all (all the possible combinations of features) and seeing how well the resulting classifier works. This may be computationally intractable, and unless a large test set is available

it may be impossible to avoid selection effect of choosing the best of a large class of classifiers on that particular test set and not for the population. To illustrate this difficulty Ripley (1996) gave the following example: Consider a battery of diagnostic tests $T_1, ..., T_m$ for a fairly rare disease, which perhaps around 5% of all patient tested actually have. Suppose test T_1 correctly picks up 99% of the real cases and has a very low false positive rate. However, there is a rare special form of the disease that T_1 cannot detect, but T_2 can, yet T_2 is inaccurate on the normal disease form. If we test the diagnostic tests one at a time, we will never even think of including T_2, yet T_1 and T_2 together may give a nearly perfect classifier by declaring a patient diseased if T_1 is positive or T_1 is negative and T_2 is positive.

2.2.2 Feature extraction

Feature extraction methods determine an appropriate subspace of dimensionality m (either in a linear or nonlinear way) in the original feature space of dimensionality d ($m < d$). Linear transforms, such as PCA, factor analysis (FA), linear discriminant analysis, and projection pursuit have been widely used in pattern recognition for feature extraction and dimensionality reduction. The best known linear feature extractor is PCA (Jain et al. 2000). The following sections are devoted for the PCA: its conventional derivation, shortcomings and recent extensions.

2.3 Principal component analysis

Principal component analysis (PCA) is one century old now. Nevertheless, it still undergoes research and new extensions are found. Pearson (1901) is the first to describe the technique. A description of practical computing methods came much later from Hotelling (1933). Later, with the advent of electronic computers, the technique achieved widespread use as the machine computational power increases the dimension of the data that can be manipulated by many order of magnitudes.

The conventional derivation of PCA is in terms of a standardized linear projection which maximizes the variance in the projected space. For a set of observed d-dimensional data vectors $\{\mathbf{y_i}\}_1^N$, the m *principal axes* $\mathbf{A_k}$, $k \in \{1, ..., m\}$ are those

orthonormal axes onto which the returned variance under projection is maximal. It is well known that the vectors $\mathbf{A_k}$ are given by the m dominant eigenvectors of the sample covariance matrix.

$$S = \sum_i (\mathbf{y}_i - \boldsymbol{\mu})^T (\mathbf{y}_i - \boldsymbol{\mu}) \tag{2.3.1}$$

Where $\boldsymbol{\mu}$ is the sample data mean. The m principal components of the observed data vector $\mathbf{y_i}$ are given by the vector $\mathbf{x_i} = \mathbf{A^T}(\mathbf{y_i} - \boldsymbol{\mu})$, where $\mathbf{A} = (\mathbf{A_1}, \mathbf{A_2}, ..., \mathbf{A_m})$. Appendix A gives a complete PCA eigenvector based definition. PCA enjoys the following important properties

i. The projection from and to the principal subspace are easy operations, since only matrix multiplications is required.

ii. Model parameters can be computed easily; e.g. from the data covariance matrix diagonalization.

iii. PCA is the optimal linear scheme for compressing, in terms of mean squared error.

From the last property emerges an important function that has been used extensively in classification, which known as *squared reconstruction error*. If $\mathbf{y}$ is a vector in the data space and $\mathbf{x}$ is its projection into the principal subspace and the orthogonal columns of $\mathbf{A}$ specify the principal subspace. We can reconstruct the data vector from its image in the principal subspace using the formula $\hat{\mathbf{y}} = \mathbf{A}\mathbf{x} + \boldsymbol{\mu}$. How far this reconstructed vector, $\hat{\mathbf{y}}$, from the original vector is the squared reconstruction error that can be defined as follows

$$D_{re}(\mathbf{y}) = \sum \| \mathbf{y} - \hat{\mathbf{y}} \|^2 \tag{2.3.2}$$

In training, a principal subspace is estimated for each class. In testing, the class that gives the least squared reconstruction error for the testing pattern is the winner.

2.3.1 Shortcomings of PCA

PCA has three main limitations

i. The technique is globally linear. If the data seems to fall in some nonlinear manifold (surface) PCA finds principal subspace that poorly fits the data. Figure 1.1 illustrates this problem graphically.

ii. A lack of a probabilistic or generative model. If we perform PCA on some data and then ask how well the model fits new data points, the only criterion available is the squared reconstruction error function given in Equation 2.3.2. A data point far away form the training data but nonetheless near the principal subspace will be assigned low squared reconstruction error. This fact is explained in Figure 2.1. Figure 2.2 shows how this shortcoming may lead to misclassification. Moreover, the lack of a generative model is a major drawback for development and testing, as it is not possible to generate synthesized data set as a simple toy to test new PCA-based methods.

iii. Naive training methods have troubles with high dimensional data and data scarcity. The conventional method that entails the calculation of the covariance matrix needs a large set of training data points for high dimensional data. For high dimensional data the calculation of the covariance data matrix is computationally intensive $\mathcal{O}(Nd^2)$, where d is dimensionality and N is training sample size (Roweis 1997).

2.4 Nonlinear PCA

To overcome the PCA global linearity limitation, many extensions for defining a nonlinear principal manifolds have been proposed in the literature (DeMers & Cottrell 1993, Hastie 1984, Hsieh 2001, Karmer 1991). The defining property of these nonlinear principal manifolds or surfaces is that their projection in the original observed space is typically a nonlinear (curved) low dimensional surface that passes through the middle of the data. This can be thought of as nonlinear regression on the data. An example of a principal curve is shown in Figure 1.1.

Autoassociative multilayer neural network, shown in Figure 2.3 is an example for neural network implementation for nonlinear principal manifolds. The middle layer

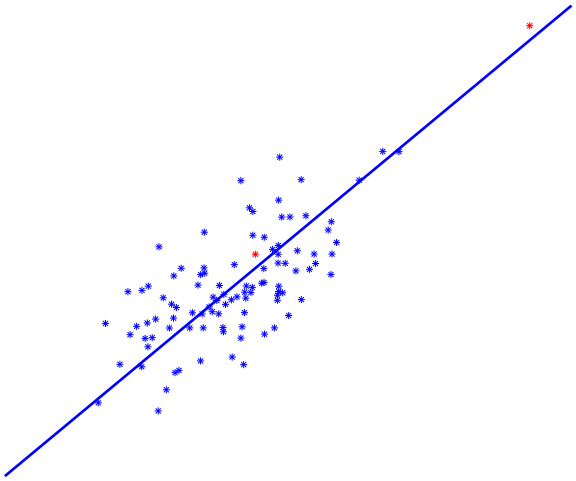


Figure 2.1: PCA does not define a proper density model in the data space; thus a point far away form the data, but nonetheless near to the principal subspace will have low reconstruction error. The two red data points will have the same reconstruction error despite the fact one of them is far from the bulk of the data.

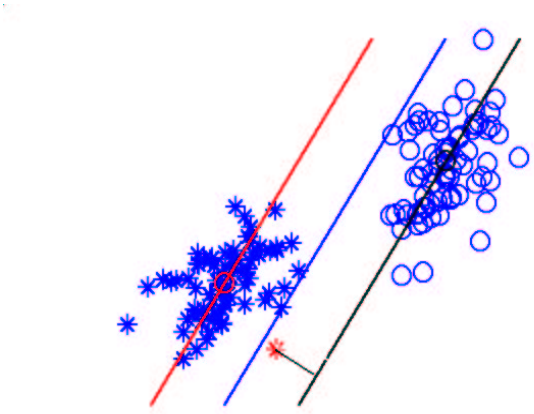


Figure 2.2: A data point that falls near to class B (appears as stars) region and has a lower reconstruction distance from class A (appears as circles) subspace than class B subspace, will be classified erroneously as belonging to class A.

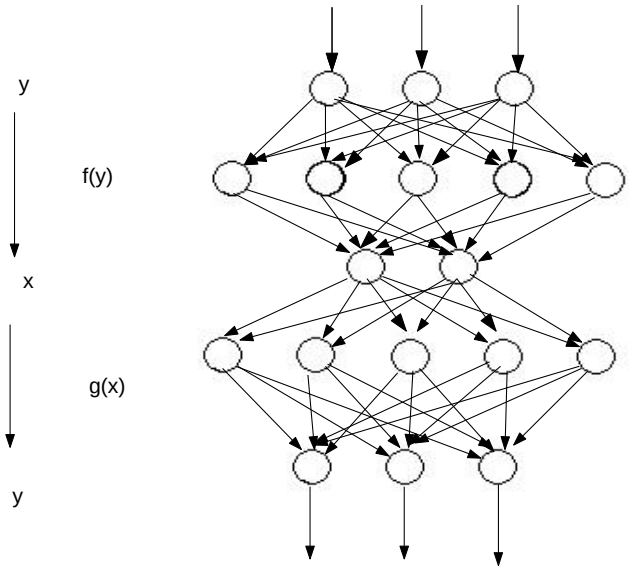


Figure 2.3: An autoassociative neural network for computing principal manifolds $\mathbf{x}$ in the input space of $\mathbf{y}$

(called "bottleneck") forms a lower-dimensional manifold representation by means of a nonlinear projection function (weighted sum of sigmoids), $f(\mathbf{y})$. The resulting principal components $\mathbf{x}$ has an inverse mapping with a similar nonlinear reconstruction function $g(\mathbf{x})$, which regenerate the input data as accurately as possible.

The current problem of all these nonlinear principal manifolds is that, these are often hard to compute in high dimensional spaces and their algorithms cannot be guaranteed to converge. This thesis follows the recent promising approach, of approximating these nonlinear manifolds using a mixtures of linear principal subspaces. Chapter 3 we give a detailed description for principal subspaces mixtures.

2.5 Probabilistic PCA

The defect of the squared reconstruction error function in Equation 2.3.2 as being unnormalized within the principal subspace led many researchers to develop probabilistic

model for PCA (henceforth PPCA). While the resulting models of those researchers are generally the same, they have different approaches. We categorize these approaches as latent-variable-based approach (Roweis 1997, Tipping & Bishop 1999*a*) and distance-based approach (Moghaddam & Pentland 1997, Sung & Poggio 1998).

2.5.1 Advantages of probabilistic PCA

Deriving PCA from the perspective of density estimation would offer a number of important advantages. Tipping & Bishop (1999*a*) summarize these advantages as follows

- The corresponding likelihood would permit comparison with other density-estimation techniques and facilitate statistical testing.

- Bayesian inference methods could be applied by combining the likelihood with a prior.

- In classification, PCA could be used to model class-conditional densities, thereby allowing the posterior probabilities of class membership to be computed.

- The value of the probability density function could be used as a measure of the "degree of novelty" of a new data point, an alternative approach to that of Petsche et al. (1996).

- The probability model would offer a methodology for obtaining a principal component projection when data values are missing.

- The single PCA model could be extended to a mixtures of such models. This advantage of particular significance for PCA mixture models as discussed in Chapter 3.

2.5.2 Latent variable approach

2.5.2.1 The latent variable model

A latent variable mode seeks to relate a d-dimensional observed data vectors $\mathbf{y}$ to a corresponding hidden m-dimensional vector of $\mathbf{x}$.

$$\mathbf{y} = f(\mathbf{x}; \mathbf{a}) + \mathbf{w} \tag{2.5.1}$$

where $f(\cdot;\cdot)$ is a function of the latent variable vector $\mathbf{x}$ with parameters $\mathbf{a}$, and $\mathbf{w}$ is independent noise process. Generally $m < d$ such that the latent variables offer more compressed description of the data. By defining a prior distribution over $\mathbf{x}$, together with the distribution of $\mathbf{w}$, Equation 2.5.1 induces a corresponding distribution in the data space, and the model parameters may then be determined by maximum likelihood techniques. Such a model may also be termed 'generative', as data vectors $\mathbf{y}$ may be generated by sampling from the $\mathbf{x}$ and $\mathbf{w}$ distributions and applying Equation 2.5.1.

2.5.2.2 PCA as a latent variable model

A linear latent variables model relates a d-dimensional observed data vector $\mathbf{y}$ to a corresponding m-dimensional vector of latent variables $\mathbf{x}$.

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \boldsymbol{\mu} + \mathbf{w} \tag{2.5.2}$$

where $\mathbf{A}$ is $d \times m$ matrix, $\boldsymbol{\mu}$ is the data mean. If both $\mathbf{x}$ and $\mathbf{w}$ are Gaussians

$$\mathbf{x} \sim N(\mathbf{0}, \mathbf{B}), \qquad \mathbf{w} \sim N(\mathbf{0}, \mathbf{Q}) \tag{2.5.3}$$

then $\mathbf{y}$ is also Gaussian given by

$$\mathbf{y} \sim N(\boldsymbol{\mu}, \mathbf{A}\mathbf{B}\mathbf{A}^{\mathbf{T}} + \mathbf{Q}) \tag{2.5.4}$$

and the model now is called linear Gaussian model. As there is a degeneracy between the structure of $\mathbf{B}$ and $\mathbf{A}$ in (2.5.4), there is no loss of generality in restricting $\mathbf{B}$ to be diagonal. Furthermore, there is an arbitrary sharing of scale between a diagonal $\mathbf{B}$ and $\mathbf{A}$: typically we either restrict the columns of $\mathbf{A}$ to be unit vectors or make $\mathbf{B}$ the identity matrix to resolve this degeneracy. In what follows, we will assume $\mathbf{B} = \mathbf{I}$

without loss of generality. Secondly, if $\mathbf{Q}$ were not restricted, learning algorithm could simply choose $\mathbf{A} = \mathbf{0}$ and then set $\mathbf{Q}$ to be the sample covariance matrix of the data, thus trivially achieving a maximum likelihood model by explaining all of the structure in the data as noise. Now, if we restrict the shape of the noise -**w**- covariance by constraining $\mathbf{Q}$, we can avoid this trivial solution and more interestingly, force desirable structure to appear in both $\mathbf{A}$ and $\mathbf{Q}$ as a result. If we restrict $\mathbf{Q}$ to be diagonal (in other words the covariance ellipsoid of $\mathbf{w}$ is axis aligned) then we recover exactly the statistical method *factor analysis* (henceforth FA). In FA the subspace defined by the columns of $\mathbf{A}$ will generally not be corresponding to the principal subspace of the data. Nevertheless certain links between FA and PCA have been noted (Basilevsky 1994, Hinton et al. 1997, Tipping & Bishop 1999*a*). If $\mathbf{Q} = \sigma^2\mathbf{I}$, then PCA emerges if the $d - m$ smallest eigenvalues of the sample covariance matrix Equation 2.3.1, are exactly equal. If this is the case, i.e., $\mathbf{B} = \mathbf{I}$ and $\mathbf{Q} = \sigma^2\mathbf{I}$ then the model covariance matrix, $\mathbf{C}$, defined as $\mathbf{ABA^T} + \mathbf{Q}$ in (2.5.4) will be reduced to

$$\mathbf{C} = \sigma^2\mathbf{I} + \mathbf{A}\mathbf{A}^T \tag{2.5.5}$$

and both $\mathbf{A}$ and σ^2 may then be determined analytically through eigen-decomposition of the data covariance matrix ,S, without resort to iteration as in FA. This is restrictive and rarely justified in practice. However, an exact covariance model for PCA is generally undesirable. In practical application of PCA, we often do not require and exact characterization of the covariance structure in the minor space, since this information is effectively discarded in the dimensionality reduction process.

After this package of fruitful restrictions, the probabilistic (generative) model (Equation 2.5.2) could be used to relate data space and principal subspace probabilistically. This allows the following definitions of probability distributions over **y**-space and **x**-space

$$p(\mathbf{y}) = (2\pi)^{-d/2}|\mathbf{C}|^{-1/2}\exp\left(-\frac{1}{2}(\mathbf{y}-\boldsymbol{\mu})^T\mathbf{C}^{-1}(\mathbf{y}-\boldsymbol{\mu})\right) \tag{2.5.6}$$

$$p(\mathbf{y}|\mathbf{x}) = (2\pi\sigma^2)^{-d/2}\exp\left(-\frac{1}{2}\sigma^2\|\mathbf{y}-\mathbf{A}\mathbf{x}-\boldsymbol{\mu}\|^2\right) \tag{2.5.7}$$

$$p(\mathbf{x}) = (2\pi)^{-m/2}\exp\left(-\frac{1}{2}\mathbf{x}^T\mathbf{x}\right) \tag{2.5.8}$$

And now we can define the log-likelihood under this model using Equation 2.5.6 for a set of observed data $\{\mathbf{y_i}\}_1^N$ as follows

$$\mathcal{L} = \sum_{i=1}^{N} \ln(p(\mathbf{y_i})) \tag{2.5.9}$$

Tipping & Bishop (1999*b*) have shown that the log-likelihood (2.5.9) is maximum when the columns of $\mathbf{A}$ span the principal subspace of the data. they have also show that the maximum likelihood estimators for $\boldsymbol{\mu}$ and σ^2 are

$$\sigma^2 = \frac{1}{d-m} \sum_{k=m+1}^{d} \lambda_k \tag{2.5.10}$$

$$\boldsymbol{\mu} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{y_i}. \tag{2.5.11}$$

where $\{\lambda_k\}_{m+1}^d$ are the $d-m$ small eigenvalues.

2.5.3 Distance-based approach

Using a normalized distance instead of the squared reconstruction error distance — which is Euclidean distance that considered normal distance for a very restricted density, namely a density with a covariance matrix equal to the identity matrix, $\mathbf{C} = \mathbf{I}$— should lead to accurate and extensible model. Assuming that the training data has Gaussian distribution, then the normalized Gaussian distance i.e., the negative log of the Gaussian formula Equation 2.5.6 should be adopted

$$D_{mh}(\mathbf{y}) = d/2 \log(2\pi) + 1/2 \log|\mathbf{C}| + 1/2(\mathbf{y}-\boldsymbol{\mu})^T \mathbf{C}^{-1}(\mathbf{y}-\boldsymbol{\mu}) \tag{2.5.12}$$

However, Moghaddam & Pentland (1997) have dropped the constant term $d/2 \log(2\pi)$ and the term $1/2 \log|\mathbf{C}|$ gaining the advantage of being saved from the ramification of the determinant of a poorly conditioned covariance matrix, $\mathbf{C}$, and the disadvantage of using not fully normalized distance. In other words, they have taken the *Mahalanobis distance* as a sufficient statistics for their derivation. If $\boldsymbol{\mu}$ is the data mean and $\tilde{\mathbf{y}} = \mathbf{y} - \boldsymbol{\mu}$, then the Mahalanobis distance could be given by

$$D_{mh}(\mathbf{y}) = \tilde{\mathbf{y}} \mathbf{C}^{-1} \tilde{\mathbf{y}} \tag{2.5.13}$$

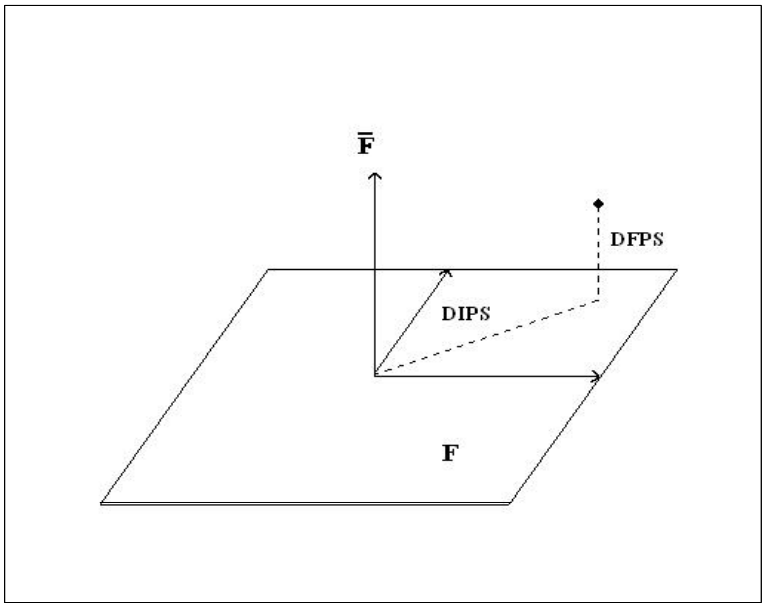


Figure 2.4: Space decomposition into Principal subspace $\mathbf{F}$ and orthogonal complement $\overline{\mathbf{F}}$

By factorizing the matrix $\mathbf{C}^{-1}$ to $\mathbf{A}\mathbf{\Lambda}^{-1}\mathbf{A}^{\mathbf{T}}$, where $\mathbf{A}$ and $\mathbf{\Lambda}$ are the eigenvectors and eigenvalues of $\mathbf{C}$ respectively.

$$\begin{aligned} D_{mh}(\mathbf{y}) &= \tilde{\mathbf{y}}^T[\mathbf{A}\mathbf{\Lambda}^{-1}\mathbf{A}^{\mathbf{T}}]\tilde{\mathbf{y}} \\ &= \mathbf{x}^{\mathbf{T}}\mathbf{\Lambda}^{-1}\mathbf{x} \end{aligned} \tag{2.5.14}$$

where $\mathbf{x} = \mathbf{A}^{\mathbf{T}}\tilde{\mathbf{y}}$ are new variables in the transformed space and since $\mathbf{\Lambda}$ is diagonal, the Mahalanobis distance can also be expressed in terms of the sum

$$D_{mh}(\mathbf{y}) = \sum_{i=1}^{d} \frac{x_i^2}{\lambda_i} \tag{2.5.15}$$

If $\mathbf{F} = \{\mathbf{A}_i\}_1^m$ is the principal subspace and $\bar{\mathbf{F}} = \{\mathbf{A}_i\}_{m+1}^d$ is its orthogonal complement, we can decompose the distance as follows

$$D_{mh}(\mathbf{y}) = \sum_{i=1}^{m} \frac{x_i^2}{\lambda_i} + \sum_{i=m+1}^{d} \frac{x_i^2}{\lambda_i} \tag{2.5.16}$$

as the second term contains minor eigenvalues that are usually ignored by the conventional PCA model and the term is summed up as reconstruction error, Moghaddam

& Pentland (1997) approximated the second term as $D_{re}(\mathbf{y})/\sigma^2$ ($D_{re}(\mathbf{y})$ is given by Equation 2.3.2) and show that the optimal value for σ^2 could be

$$\sigma^2 = \frac{1}{d-m} \sum_{i=m+1}^{d} \lambda_i \tag{2.5.17}$$

i.e. the average of the eigenvalues outside the principal subspace and Equation 2.5.16 is now

$$D_{mh}(\mathbf{y}) = \sum_{i=1}^{m} \frac{x_i^2}{\lambda_i} + D_{re}(\mathbf{y})/\sigma^2 \tag{2.5.18}$$

The first factor in Equation 2.5.18 is named as Distance-inside-principal-subspace (DIPS), and the second is named Distance-from-principal-subspace (DFPS). Figure 2.4 illustrate this decomposition graphically. Using Equation 2.5.18 we can approximate the Gaussian density function as a product of two marginal densities as follows

$$\hat{p}(\mathbf{y}) = p_F(\mathbf{y})\hat{p}_{\overline{F}}(\mathbf{y}) \tag{2.5.19}$$

where $p_F(\mathbf{y})$ is the true marginal density in the principal subspace F-space and $\hat{p}_{\overline{F}}(\mathbf{y})$ is estimated marginal density in the orthogonal complement $\bar{F}$-space. All probabilistic reasonings on PCA are now forward using Equation 2.5.19.

2.6 Conclusions

PCA is the most widely used feature extraction method (Jain et al. 2000). PCA generates uncorrelated, optimal-reconstruction-error subspaces. To widen its applicability even more, solutions for its shortcomings should be found. PCA probabilistic formulation avails all the advantages listed in Section 2.5.1. The price we pay for adopting the probabilistic PCA is that we restrict the model to Gaussian data. In Chapter 3 we explain how this price payed back by using the mixture of Gaussians —rather than one Gaussian— which can model arbitrarily distributed data.

Recent studies show that mixture of local linear subspace models is faster and more accurate than nonlinear PCA, specially for high dimensional data (Kambhatla 1995, Moghaddam & Pentland 1997). For this trend, the probabilistic definition of PCA is very important as it facilitates formal way for defining and analyzing the PCA mixtures model. This is the main topic of Chapter 3.

CHAPTER 3

Mixture of local principal subspaces

3.1 Introduction

In Chapter 2, the PCA main characteristics are explained with spacial emphasis on the recent development of defining probabilistic model for PCA. As the major shortcoming of PCA is its global linearity, in Section 2.4 we give an overview for the recent major research done in order to formulate nonlinear PCA as a search for a curved manifold (as opposed to linear subspace) that is close to the data. This thesis, follows the alternative paradigm of capturing the data complexity and heterogeneity by a mixture of local linear subspaces model. The idea is that, when it is difficult to find a global linear subspace that fit the data, we may find a set of local linear principal subspaces that fit the data in its local regions. As an illustration, we produce two figures for a 256-dimensional digit "2" data set, Figure 3.1 and Figure 3.2. The first part of Figure 3.1 shows a global 1-dimensional principal subspace for the given data set. The middle image in this row of images is the mean, i.e., the origin of the subspace. The last image is the principal vector of the subspace (designated with name 1.PC). The other ten images, five on left of the mean and five on right of the mean are inverse images for points on this 1-dimensional subspace at a multiple of constant distance from the mean. The same data used in generating the global principal subspace is partitioned into 10 clusters. For each one of these 10 clusters a local subspace is generated. The second part of Figure 3.1 illustrates these local subspaces in the same

manner as in the first part. While the images in the first part look noisy and central to digit "2" data space, the images in the second part characterize local regions of digit "2" data space. Also, it is notable the principal vector for the global subspace tend to be more general while the principal vectors of the local subspaces tend to specify some of the different shapes exist in digit "2" data space.

Figure 3.2 shows inverse images from a 2-dimensional global data space. If one compares these two figures, he can easily notice that: Increasing the dimensionality of the global subspace fails dismally in producing better fit. Even with two dimensions the subspace is still located in a central region in this high dimensional space and its inverse images look very noisy, indicating that the subspace do not fit the data well. Therefore, this global subspace fails to capture the different structures located in regions far from the data space central region.

This chapter deals with formalizing and designing an autonomous scheme for finding these local principal subspaces. This is an unsupervised problem in which we want to find the optimal number of clusters in the given training data. In addition, we need also to find the optimal dimensionality for each local principal subspace.

In this chapter, we introduce clustering methods. Section 3.3 explains the difference between *hard clustering* and *soft clustering* and shows that the former is a limiting case of the latter. Section 3.4 defines the mixture model. This section also explains the most widely used mixture model, namely, the Gaussian mixture model (henceforth MoG). Section 3.5 explains the notion of subspace clustering. As PPCA is a Gaussian model, in Section 3.5.2 we define the mixture of PPCA (henceforth MPPCA). We then define a new greedy learning scheme in section 3.6, that autonomously finds a suboptimal number of principal subspaces in the maximum likelihood framework. The scheme finds subspaces with different dimensionalities and equal local variability. The scheme also contains a new method for initializing the EM for MPPCA.

3.2 Issues in clustering

As there are many methods now for clustering, the term became a generic label for a variety of procedures designed to find natural groupings, or clusters, in multidimen-

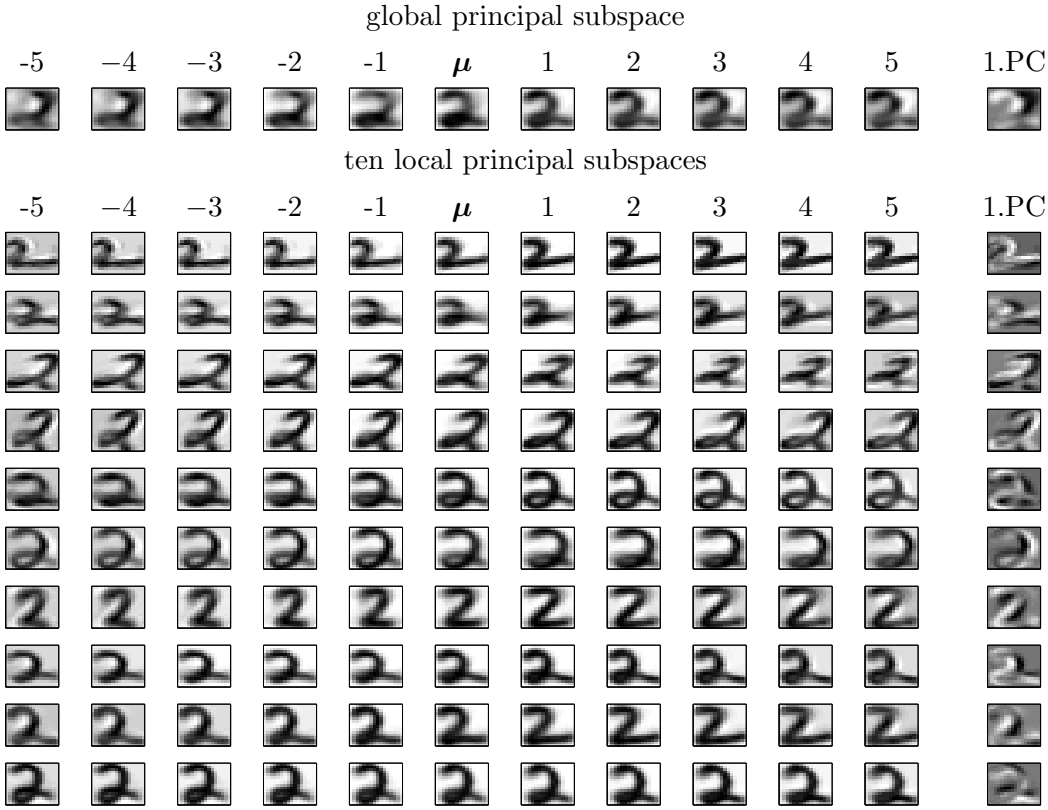


Figure 3.1: Illustration of how 1-dimensional local principal subspaces capture different structure of digit "2" high dimensional space (256 dimensions).

sional data, based on measured or perceived similarities. Several clustering algorithms have been proposed in the literature and new clustering algorithms continue to appear (Hartigan 1975, Jain & Dubes 1988, Kaufman & Rousseeuw 1990). Clustering is a very difficult problem because data can reveal clusters with different shape and sizes. In general, a complete cluster scheme should deal with the following three basic issues:

- How to estimate the number of clusters in the given training data.
- How to find a suitable clustering criterion for the given problem.
- How to estimate the parameters defining each cluster.

In some problems, we may not need to deal with the first issue, as the number of clusters may be known beforehand. However, for some practical reasons we couldn't

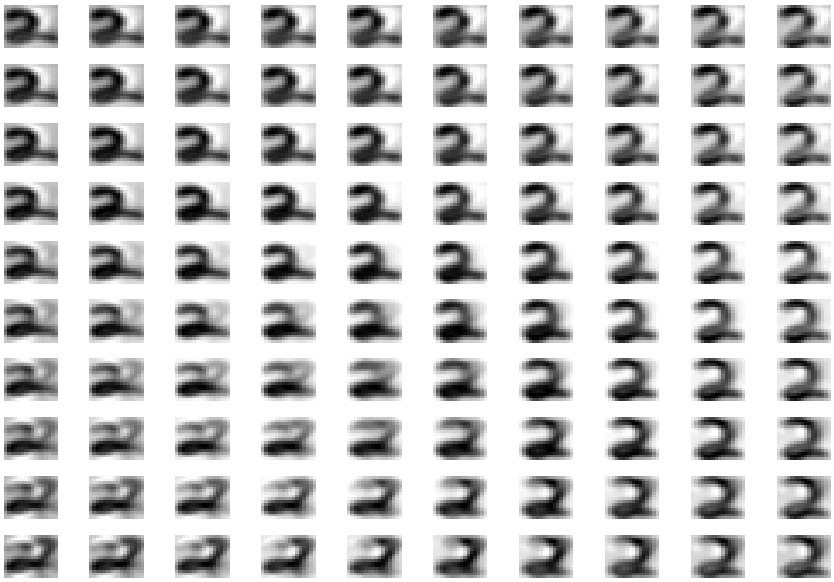


Figure 3.2: Images generate from points in a 2-dimensional global principal subspace for digit "2" the original data space dimensionality is 256.

label the training data set. Dealing with unknown number of clusters problem is more complex. This case is more common if we have a multi-mode class data set. Which is type of problems considered in this thesis.

Its worth here, to mention the issues, that Jain et al. (2000) advice the clustering algorithm user to keep in mind

i Every clustering algorithm will find clusters in a given dataset whether they exist or not.

ii There is no best clustering algorithm. Therefore a user is advised to try several clustering algorithms on a given dataset.

iii Issues of data collection, data representation, normalization, and cluster validity are as important as the choice of clustering strategy.

3.3 Hard clustering versus soft clustering

Clustering could be categorized as *hard clustering* and *soft clustering*. The objective of a hard clustering training is to define clusters in the given training data and to

assign each testing data point to only one cluster. We can view hard clustering as a boolean function that takes as input a data point and a cluster label. For a given data point the function returns true for only one cluster label and false for all other cluster labels. On the other hand, the objective of a soft clustering method is to produce a set of density functions that assign to a given data point probabilistic values for its belongingness to each cluster. In general, in the literature the term clustering is used for hard clustering and the term *mixture models* is used for soft clustering.

Hard Clustering entails assigning each data point in an unlabelled data set to exactly one of k possible classes or *clusters* so that the points in each given cluster are similar to one another. A loss function that penalizes dissimilar points within a cluster is used in training. A commonly used hard clustering method is the k-means algorithm (Kaufman & Rousseeuw 1990, Jain & Dubes 1988). It assigns points at random to k classes and then computes the mean vector for each class. Points are then reassigned to the classes with the nearest mean and the means are recomputed. This is repeated until convergence. While hard clustering algorithm are often computationally efficient, they fail to define a probability density function. Hard Clustering can be thought of as a special case of a soft clustering that always gives one for one cluster and zero for the other clusters. In fact, if the soft clustering method defines a mixture of Gaussians with variances tending to zero, then it is effectively a hard clustering. While hard clustering algorithms fail to define a probability density functions, they are often computationally efficient and indeed provide useful initialization for soft clustering methods. For illustration purposes, Figure 3.3 shows scatter diagrams for two data sets. Figure 3.3a contains a relatively separated clusters, for which hard clustering methods can give good result as shown in Figure 3.3b. The clusters shown in Figure 3.3c are interfering. For such types of clusters hard clustering methods gives different clustering results, especially for boundary points. Figure 3.3d shows the results for training a mixture of Gaussians (see Section 3.4.1) on this data set.

3.4 Mixture models (soft clustering)

Mixture models are flexible and powerful probabilistic modelling tools. Mixtures offer formal (i.e. model-based) approaches to clustering. Mixtures adequately model situations where each pattern is produced by one of a set of alternative (probabilistically modelled) sources (Titterington et al. 1985, McLachlan & Peel 2000). Nevertheless, it should be kept in mind that strict adherence to this interpretation is not required; mixture can also be seen as a class of models that are able to represent arbitrarily complex probability density functions.

If $p_j(\mathbf{y};\boldsymbol{\theta}_\mathbf{j})$ is the j-th component parameterized on $\boldsymbol{\theta}_\mathbf{j}$, then a mixture density of a random vector $\mathbf{y}$ assuming k components is

$$f_k(\mathbf{y}|\boldsymbol{\theta}) = \sum_{j=1}^{k} q_j p_j(\mathbf{y};\boldsymbol{\theta}_\mathbf{j}) \tag{3.4.1}$$

where $\boldsymbol{\theta}$ comprises all parameters of the component densities $p_j(\mathbf{y};\boldsymbol{\theta}_\mathbf{j})$ and the mixing weights q_j which satisfy $\sum q_j = 1$ and $q_j \geq 0$.

3.4.1 Mixture of Gaussians

While components of a mixture could be non-Gaussian, Gaussian-based mixture density model has proven to be the most exceedingly popular mixture density model; perhaps mainly due to its simplicity, general applicability and extendibility. A brief history of the many uses of MoGs, spanning fields as varied as psychology, geology, and astrophysics, has been compiled by Titterington et al. (1985). According to the central limit theorem, the distribution of a sum of two independent random variables usually has a distribution that is closer to Gaussian than any of the original random variables. Therefore, in many situations, it is easier to find artificial Gaussian mixture components than arriving at the intrinsic original non-Gaussian components, if they exist. This is frustrating, if our practical goal is to find the intrinsic components. However, in classification problems one seeks higher classification accuracy rather than finding intrinsic components, i.e. artificial Gaussian components that increase classification accuracy are highly appreciated. Furthermore, the main advantage of Gaussian mixture modelling, is that given enough mixture components we can model (almost)

any density (as accurately as desired). Figure 3.4 reveals this fact.

A MoG is given by the weighted sum as in Equation 3.4.1, where the j-th component $p_j(\mathbf{y};\theta_j)$ is a d-dimensional Gaussian density:

$$p(\mathbf{y};\boldsymbol{\theta_j}) = (2\pi)^{-\frac{d}{2}}|\mathbf{S_j}|^{-\frac{1}{2}}\exp\left(-\frac{1}{2}(\mathbf{y}-\boldsymbol{\mu_j})^T\mathbf{S_j}^{-1}(\mathbf{y}-\boldsymbol{\mu_j})\right) \tag{3.4.2}$$

parameterized on the mean, $\boldsymbol{\mu_j} \in \mathbb{R}^d$, and the covariance matrix, $\mathbf{S_j}$, collectively denoted by the parameter vector $\boldsymbol{\theta_j}$. For the learning problem, we assume a training set $\{\mathbf{y}\}_1^N$ data points, and the task is to estimate the parameters $\{q_j, \boldsymbol{\mu}_j, \mathbf{S}_j\}$ of the k components that maximize the log-likelihood:

$$\mathcal{L} = \sum_{i=1}^{N}\log(f_k(\mathbf{y_i})) \tag{3.4.3}$$

where $f_k(\mathbf{y_i})$ is defined by Equation 3.4.1

3.4.2 Mixture of Gaussians training

For a unimodal model (i.e., the mixture has only one maxima and therefore, one component which is by no way a mixture), the maximum likelihood framework offers direct method for model parameters estimation (Schalkoff 1992).

However, direct method for multimodal (i.e., the mixture has more than one maxima and therefore, more than one component) does not exist. To illustrate this, consider a MoG with covariance matrices of the restricted form, $\mathbf{S_j} = \sigma_j^2\mathbf{I}$ where $\mathbf{I}$ is the identity matrix, which results in the following definition for the density function

$$p(\mathbf{y};\boldsymbol{\theta_j}) = (2\pi\sigma_j^2)^{-\frac{d}{2}}\exp\left\{-\frac{||\mathbf{y}-\boldsymbol{\mu_j}||^2}{2\sigma_j^2}\right\}$$

To find maximum likelihood estimation for parameter vector $\boldsymbol{\theta_j}$. We can minimize the negative log-likelihood function, which can be regarded as an error function

$$\mathcal{E} = -\mathcal{L} = -\sum_{i=1}^{N}\log(\sum_{j=1}^{k}q_j p_j(\mathbf{y};\boldsymbol{\theta_j})) \tag{3.4.4}$$

with respect to the parameters $\boldsymbol{\mu_j}, \sigma^2$ and q_j. Assume the only unknowns are the mean vectors, $\boldsymbol{\mu_j}, j = 1, 2, ...k$. Thus

$$\boldsymbol{\theta} = \{\boldsymbol{\mu_j} : j = 1, 2, ...k.\} \tag{3.4.5}$$

Now, by differentiating $\mathcal{E}$ with respect to $\boldsymbol{\mu}_\mathbf{j}$ we get

$$\frac{\partial \mathcal{E}}{\partial \boldsymbol{\mu}_\mathbf{j}} = \sum_{n=1}^{N} P(j|\mathbf{y^n}; \boldsymbol{\theta}_\mathbf{j}) \frac{(\mathbf{y^n} - \boldsymbol{\mu}_\mathbf{j})}{2\sigma_j^2} \tag{3.4.6}$$

and then by setting to zero we obtain

$$\hat{\boldsymbol{\mu}}_\mathbf{j} = \frac{\sum_n P(j|\mathbf{y^n}; \boldsymbol{\theta}_\mathbf{j}) \mathbf{y^n}}{\sum_n P(j|\mathbf{y^n}; \boldsymbol{\theta}_\mathbf{j})} \tag{3.4.7}$$

Note that $\boldsymbol{\mu}_\mathbf{j}$ implicitly occurs in righthand side of Equation 3.4.7. As the term $P(j|\mathbf{y^n}; \boldsymbol{\theta}_\mathbf{j})$ is the posterior probability for component j given the data point $\mathbf{y^n}$, which can be expressed using Bayes' theorem as follows

$$P(j|\mathbf{y}; \boldsymbol{\theta}_\mathbf{j}) = \frac{q_j p(\mathbf{y}|j; \boldsymbol{\theta}_\mathbf{j})}{p(\mathbf{y}; \boldsymbol{\theta}_\mathbf{j})} \tag{3.4.8}$$

We cannot estimate the parameters from this set of highly non-linear coupled equations directly. However, there are three approaches for dealing with this problem (Render & Walker 1984). One approach is to employ the standard non-linear optimization techniques (Bishop 1995). The second is a stochastic technique, based on the assumption that the data point are arriving one at a time, and we have a sequential update scheme that uses the data points one after the other to update the parameters (Traven 1991). The third approach, is based on the expectation maximization algorithm. The expectation maximization algorithm has been by far the most commonly used approach (McLachlan & Peel 2000) for fitting the MoG. Hence the focus in this thesis is on fitting of MoG via the expectation maximization algorithm.

3.4.3 Expectation maximization algorithm

The Expectation Maximization algorithm (henceforth EM) is an iterative algorithm for learning mixture density by attempting to maximize the likelihood. EM consists of two main steps: expectation step (*E-step*) and maximization step (*M-step*). The algorithm starts with some random initial values for the unknown parameters vector $\boldsymbol{\theta}$. The E-step computes the posteriori probabilities for the training data set using the given parameters set and hence the likelihood. The M-step then finds new values for $\boldsymbol{\theta}$ that maximize the likelihood. The algorithm iterates between these two steps until

Algorithm 1 Expectation Maximization Algorithm

initialize $\boldsymbol{\theta_0}$, T, i = 0

repeat

i = i +1

E-step compute $f_k(\mathbf{y}|\theta_i)$

M-step $\boldsymbol{\theta_{i+1}} = \arg\max f_k(\mathbf{y}|\theta_i)$

until $f_k(\theta_{i+1}) - f_k(\theta_i) < T$

RETURN $\hat{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta^{i+1}}$

a given criterion is satisfied. Algorithm 1 summarizes the EM algorithm. Appendix B gives a detailed account for the algorithm including its convergence properties.

Aitkin & Aitkin (1994) noted that almost all the post-1978 (the year of EM introduction) applications of mixture modelling reported in the books on mixtures by Titterington et al. (1985) and McLachlan & Basford (1988) use the EM. This is again revealed in the recent book by McLachlan & Peel (2000) and most recent papers (Figueiredo & Jain 2002). The study of fitting finite mixture models by maximum likelihood appeared in many papers during 1960s and 1970s (Day 1969, Wolfe 1970). However, it was the publication of the seminal paper of Dempster et al. (1977) on the EM algorithm that greatly stimulated interest in the use of finite mixture distributions to model heterogeneous data.

3.4.3.1 EM for MoG

As stated in Section 3.4.3, the algorithm starts with some initial values for the unknown parameters vector $\boldsymbol{\theta}$, i.e., $\boldsymbol{\mu_j}$, $\mathbf{S_j}$ and the weight q_j for the MoG. Using these values, the algorithm computes the posteriori probabilities for the training data using Equation 3.4.9 (E-step). In the second step the algorithm uses the recently calculated posteriori probabilities and Equation 3.4.10, Equation 3.4.11 and Equation 3.4.12 to calculate new estimation for the parameters vector $\boldsymbol{\theta}$. The calculation then, cycles from expectation to maximization and from maximization to expectation, until the revised estimate do not differ appreciably from the estimate obtained in the previous iteration. Or alternatively, until some criterion met.

- **E-step**:

$$P(j|\mathbf{y_i}) = \frac{q_j p(\mathbf{y_i}; \boldsymbol{\theta_j})}{f_k(\mathbf{y_i})} \tag{3.4.9}$$

- **M-step**:

$$q_j = \frac{1}{n}\sum_{i=1}^{n} P(j|\mathbf{y_i}), \tag{3.4.10}$$

$$\boldsymbol{\mu_j} = \frac{\sum_{i=1}^{n} P(j|\mathbf{y_i})\mathbf{y_i}}{\sum_{i=1}^{n} P(j|\mathbf{y_i})}, \tag{3.4.11}$$

$$\mathbf{S_j} = \frac{\sum_{i=1}^{n} P(j|\mathbf{y_i})(\mathbf{y_i} - \boldsymbol{\mu_j})(\mathbf{y_i} - \boldsymbol{\mu_j})^T}{\sum_{i=1}^{n} P(j|\mathbf{y_i})}. \tag{3.4.12}$$

3.4.3.2 Learning MoG with EM Difficulties

Learning a MoG using EM algorithm can be difficult because of the following reasons

- The true number of mixing components is usually unknown, and the algorithm does not offer a proper accepted method to find this number. In fact as stated by Jain et al. (2000) this is the most difficult problem in mixture modelling.
- EM algorithm can get stuck in one of the many local maxima of the likelihood function.
- There is no generally accepted method of parameter initialization.

3.5 Subspace clustering

So for we have explained the k-means algorithm as the widely used clustering algorithm. Generally, k-means offers only an estimate for the means of the clusters found. However MoG overcomes the k-means deficiency by offering more information about feature variability and correlation. The price we paid for this flexibility is the increase of the parameters to be calculated. The main disadvantage of the MoG is the large number of parameters needed, $\mathcal{O}(k \times d^2)$, where k is the number of components in the

mixture and d is the dimensionality. This problem promotes the subspace clustering as a solution that goes beyond modelling clusters merely by their means, as well as having far less parameters than what is needed for a MoG.

The simplest way of clustering data in subspaces is to use a variation on the k-means algorithm (Devijver & Kittler 1982). The main difference with k-means algorithm is that, while k-means only recalculate the cluster means in each iteration, the k-subspaces algorithm recalculates the PCA projection matrix $\mathbf{A}$ as well. Furthermore, the distance measure used is a distance to a subspace e.g., Equation 2.3.2 rather than a distance to a cluster center only. Almost all other subspace clustering methods follow a similar scenario, i.e., integrating a clustering method like the k-means with a dimensionality reduction method like the PCA.

3.5.1 Mixtures of PCA

A number of "mixtures of Principal Component Analyzers" (henceforth MPCA) have been proposed in the literature before the mixtures of Probabilistic Principal Component Analyzers (henceforth MPPCA). These methods necessitate a two-stage procedure: partitioning of the data space by a hard clustering algorithm followed by the estimation the of principal subspaces within each partition (Sung & Poggio 1998, Hinton et al. 1997, Kambhatla 1995). Hinton et al. (1997) have proposed an important turning model among these early studies. They considered the partition assignments as missing data in an expectation maximization (EM) framework, and thereby propose a soft clustering algorithm where instead of any given data point being assigned exclusively to one principal component analyzer, the responsibility of its generation is shared among all of the analyzers. The authors concede the absence of a probability model for PCA is a limitation to their approach and propose that the responsibility of the j-th analyzers for reconstructing data point $\mathbf{y_i}$ can be given by $r_{ij} = \exp(E_j^2/2\sigma^2)/\{\sum_{j'} \exp(-E_{j'}^2/2\sigma^2)\}$, where E_j is the corresponding reconstruction cost. This allows the model to be determined by the maximization of a pseudo-likelihood function, and explicit two-stage algorithm is unnecessary. Unfortunately, this also requires the introduction of a variance parameter σ^2 whose value is somewhat arbitrary and again no probability density is defined.

3.5.2 Mixtures of PPCA

Probabilistic PCA model facilitates the usage of the set of MoG equations given in Section 3.4.3.1 to train a mixture of PPCA model. This can be achieved by calculating eigenvectors of the covariance matrices, $\mathbf{S_j}$, found by Equation 3.4.12 to calculate $\mathbf{A_j}$. σ_j^2 can be calculated by Equation 2.5.10 from the eigenvalues found. However, this procedure is computationally intensive as it necessitates the calculation of the data covariance matrix for each subspace and faster methods that can find the principal components (i.e. $\mathbf{A_j}$) without calculating the full covariance matrix using the EM algorithm have been described in the literature (Roweis 1997, Tipping & Bishop 1999*a*). Appendix C contains the formulas that we use in fitting the MPPCA which are found by Tipping & Bishop (1999*a*).

3.6 A complete greedy scheme for MPPCA training

As discussed in the previous sections and summarized in Section 1.2, in training an MPPCA model, the user faces a number of choices: the number of subspaces to look for, the number of dimensions per subspace to use and the way in which the EM algorithm is initialized, to avoid local minima. In this section we describe some algorithms —which form together a greedy scheme for MPPCA training— which alleviate these choices. Firstly, in Section 3.6.1 we illustrate our method for specifying a variable local dimensionality m for each subspace by using a constant local variability α in all subspace. Secondly, in Section 3.6.2 a greedy algorithm is discussed, which can be used to determine a suboptimal number of variable dimensionality subspaces, by incorporating the method described in Section 3.6.1. The last section (Section 3.6.3) describes an EM initialization method which helps in preventing local minima. These algorithms lead to an almost parameter-free MPPCA training method. The only parameter left is the amount of variance to retain α— which used in choosing the local dimensionality for each subspace.

3.6.1 Subspace dimensionality versus variability

In conventional PCA techniques, there are two approaches for specifying principal subspace dimensionality : [1] to be specified by the user beforehand. [2] the user can specify a retained variance value α, and the system calculates the dimensionality that retains this variability ratio by finding the first m eigenvectors that satisfy the following inequality

$$\Sigma_{i=1}^{m}\lambda_i/\Sigma_{i=1}^{d}\lambda_i > \alpha \tag{3.6.1}$$

Almost, all of the MPCA models described in the literature use the first approach. Most of the early proposal was geometrically based model, this could be the main reason behind the choice of the first approach. As the model shifted to be probabilistically based with Tipping & Bishop (1999*b*), Roweis (1997), Moghaddam & Pentland (1997) extensions, we think the choice in this issue should also shift to the second approach. Moreover, there are many advantages in adopting the second approach for PPCA model

- In many application fields, it seems more natural to look for a constant variability ratio to be retained by all subspaces, rather than looking for a constant dimensionality for all subspaces. For instance, in image compression, reconstructed images from same-variability subspaces are expected to be more similar than images reconstructed from same-dimensionality and different-variability subspaces.

- Intuitively, the PPCA subspace model partitions the input space into subspace of signal (principal subspace) and noise (the orthogonal complement of the minor variability directions, see Figure 2.4). Since the noise is estimated as the average of eigenvalues in the minor $d-m$ dimensions (Equation 2.5.10), there is a trade-off between subspace dimensionality and noise level estimation. Table 3.1 shows the noise level (σ^2), the classification error and the retained variance value (α) for one of our experiments on handwritten digit classification. In these experiments the dimensionality of all subspaces in fixed to ten (Musa et al. 2001*b*). Three important remarks can be drawn from this table:[1] fixing the dimensionality in all subspaces results in variant α and σ^2 in the different classes [2] σ^2 is inversely proportional to α. [3] when σ^2 value is too small for specific class this

class may suffer overfitting problem, see digit "1" (this problem is discussed in Section 4.2.1 in more details). From these experiments, we concluded that *Since σ^2 is inversely proportional to α a retained variance based model can be more robust against overfitting by avoiding very high α values.*

Meinicke & Ritter (2001) have given a similar suggestion. In their work, instead of using a constant variability ratio they have used a constant noise value. they argued that "A PPCA model that has a hypothesized noise level and estimated dimensionality is advantageous over the one that has a hypothesized fixed dimensionality and estimated noise". Meinicke & Ritter have shown that for a fixed noise level model, variable dimension PPCA, the maximum likelihood estimation of the principal subspace dimensionality is given by

$$\hat{m} = |\{\lambda_i : \lambda_i > \sigma^2_{fxd}, i = 1, ..., d\}| \tag{3.6.2}$$

where σ^2_{fxd} is a hypothesized constant value. In contrast with σ^2, σ^2_{fxd} is approximately equal to λ_{m+1}.

In our MPPCA training sechme, we fixed α value and adjust all subspaces dimensionalities to retain this given variability ratio and hence we call this model VD-MPPCA, VD for Variable Dimensionality. Moreover, as one of our primary goal is to make the model training autonomous, the adoption for this retained variance approach will be of much help in this direction. The simplest way to find α autonomously is by validation methods (Duda et al. 2000). This may be computationally intensive, however, exploitation of other speeding up methods like using some prior information to limit the choices is open question.

3.6.2 Greedy EM for MPPCA training

Recent theoretical developments (Cadez & Smyth 2000, Lee & Barron 2000), have shown that the log-likelihood of finite mixture models is approximately concave as a function of k (number of components). Accordingly, we have developed and tested a greedy EM for MPPCA training controlled by likelihood changes (Musa et al. 2001*a*). As the model used in the experiments reported in Musa et al. (2001*a*) is a greedy FD-

Table 3.1: The average σ^2, average retained variance α and average test error for each of the 10 digit classes. This table is taken form the experiments reported in Musa et al. (2001*a*)

Digit	Avg. σ^2	Avg. α	Error %
0	0.18	0.73	1.99
1	**0.06**	**0.91**	**4.99**
2	0.25	0.65	0.79
3	0.22	0.66	2.79
4	0.20	0.73	1.48
5	0.24	0.63	1.75
6	0.16	0.65	1.92
7	0.14	0.74	3.00
8	0.22	0.65	2.36
9	0.13	0.75	4.10

PPCA (Fixed Dimensionality) model, and its performance is encouraging, we extend it to be a greedy VD-PPCA for the reasons discussed above.

In addition to the encouraging result of our previous experiments, Verbeek et al. (2003) have also investigated the problem of learning MoG using a greedy EM algorithm. Finding good candidates for a new component is a primary concern in their work. Since there are many local maxima in the data space, when searching for a new component candidate, they have proposed a method for finding $k \times h$ candidates. First the data is partitioned into k partitions, related to the current k Gaussians, and from each partition h candidates are generated. To choose the best candidate they run the EM algorithm partially for each candidate (by partially we mean updating only the new candidate parameters). The candidate that generates the highest likelihood is the winner. Their experimental results showed that their greedy EM outperforms standard EM. Our approach in finding new candidate is totaly different. In our method we assume that the current k components represent part of the training data adequately and therefore we want to avoid having the new candidate located in this well repre-

sented data points. Our scheme uses the membership (i.e., the posterior probability) to the current components as a criterion for this well representation. By removing the points which has higher level of membership to the current components, we arrive at small subset of points. In this smaller subset of points we use a specialized algorithm, Algorithm 3, described in Section 3.3, to find a new candidate.

Algorithm 2 outline our new greedy VD-PPCA training algorithm. In step 2 the algorithm start by calculating a global principal subspace from the given training data set D by direct eigenvector method. The input α value is used to select the first m eigenvectors that retains α ratio from the global variability. In steps 6 we collect the points that has higher membership to the recently found subspaces $\cup N_i, i = 1..k$. For each subspace i, N_i contains the D/k points which have got the highest posterior probability for this subspace. Step 7 calculates F as the complement of the set $\cup N_i$ in the training data set D, i.e., now F contains the points that have lowest membership to the current k subspaces. In step 8, Algorithm 3 finds a new Gaussian in F. Form this new Gaussian the algorithm calculate PPCA parameters with the dimensionality determined by α and the local points (This is illustrated in Section 3.6.3). A subspace may change its shape during EM iterations, i.e., the membership of the points change form iteration to another and this results in change in the local variability and hence PCs directions. For this reason, after locating all subspaces we re-calculate subspaces' dimensionalities (step 11) and re-run the EM again (step 12) to ensure soft fitting. With the addition of new subspaces, normally old subspaces lose some of their points and hence become more local. Therefore, the process of re-estimating subspace dimensionality in step 10 normally reduces subspaces' dimensionalities.

3.6.3 Greedy hard clustering

An EM algorithm (soft clustering) initialized by a hard clustering algorithm is the state of the art for mixture model training. However, which hard clustering method and how to engage it with the EM algorithm is still an open question. However, a suitable hard clustering method for a greedy EM might also be a greedy one. Among the recently proposed Gaussian centers finding algorithm we found an algorithm that works greedily, proposed by Dasgupta Dasgupta (1999). Dasgupta showed that to

Algorithm 2 Greedy Training Algorithm

1: input D (training data set) and α (ratio of variance to retain)
2: calculate one global MPPCA subspace with a subspace dimensionality that retains α of the variability in D
3: $k \leftarrow 1$.
4: **repeat**
5: $k \leftarrow k + 1$.
6: $\forall i, N_i = \{y\text{: } y$ among the $|D|/k$ points that have highest probabilities for subspace $i\}$.
7: $F = D \backslash (\cup N_i)$
8: find a new subspace in F by algorithm 3
9: fit the k subspaces using the EM algorithm
10: **until** likelihood$(k-1) \cong$ likelihood(k)
11: re-fit all subspace dimensionalities to retain α ratio of their local variability
12: fit the model using the EM algorithm

estimate the means of a Gaussian mixture with a common covariance matrix, we can map the data to a random subspace of size $\mathcal{O}(\log k)$ dimensions, where k is the number of Gaussians, without collapsing the Gaussians together. In the reduced subspace the Gaussians become more spherical and the number of training data points with respect to the reduced subspace dimensionality is still relatively high. More importantly there is no danger of collapsing Gaussians together, as long as we project to a subspace of size $\mathcal{O}(\log k)$ (Dasgupta 1999). The algorithm allocates the Gaussians one after the other. A point with the smallest radius $r_{\mathbf{x}}$ to enclose specific number of points, p, in the projected space, is considered a clue for a Gaussian center. To find the other Gaussian centers using the same criterion, the high-density points of the recently found Gaussian are removed, this step is done by Algorithm 2 in our scheme. Moreover, as our interest is to find one Gaussian to update our greedy EM, we reduce the algorithm to allocate one Gaussian center. For the newly allocated Gaussian center and the old k Gaussians, we run the nearest mean classification algorithm to find the set S' which contains the data points nearest to the new Gaussian center. From S' we calculate

the initial estimation for the PPCA subspace parameters. Algorithm 3 describe the proposed algorithm in an algorithmic fashion.

Note that the parameter k is determined by the current stage of algorithm 2. Therefore the dimensionality of the projection subspace increases with k, and the number of points within radius $r_{\mathbf{x}}$ decreases with k. For the experiments reported in this thesis we use a projection space of $k+1$ dimensions, which is found by the first $k+1$ eigenvectors.

Algorithm 3 Initialization Algorithm: Greedy Hard Clustering Algorithm

1: input F (training data set), $\{\boldsymbol{\mu}^i\}_1^k$ (existing subspace means) and α
2: project the whole data set F into a random subspace of $\mathcal{O}(\log k)$ dimensions, let S = projected data
3: set $p \leftarrow |S|/k$
4: $\forall \mathbf{x} \in S$, let $r_{\mathbf{x}}$ be the smallest radius such that there are $\geq p$ points within distance $r_{\mathbf{x}}$ from $\mathbf{x}$
5: let $\boldsymbol{\mu}^*$ be the point $\mathbf{x}$ with the lowest radius $r_{\mathbf{x}}$
6: run the nearest mean algorithm for $\boldsymbol{\mu}^*$ and the existing k means ($\{\boldsymbol{\mu}^i\}_1^k$) in original data space to find the closest points to $\boldsymbol{\mu}^*$ (S')
7: from S' estimate one PPCA subspace parameters, with a subspace dimensionality that retains α of the variability in S'

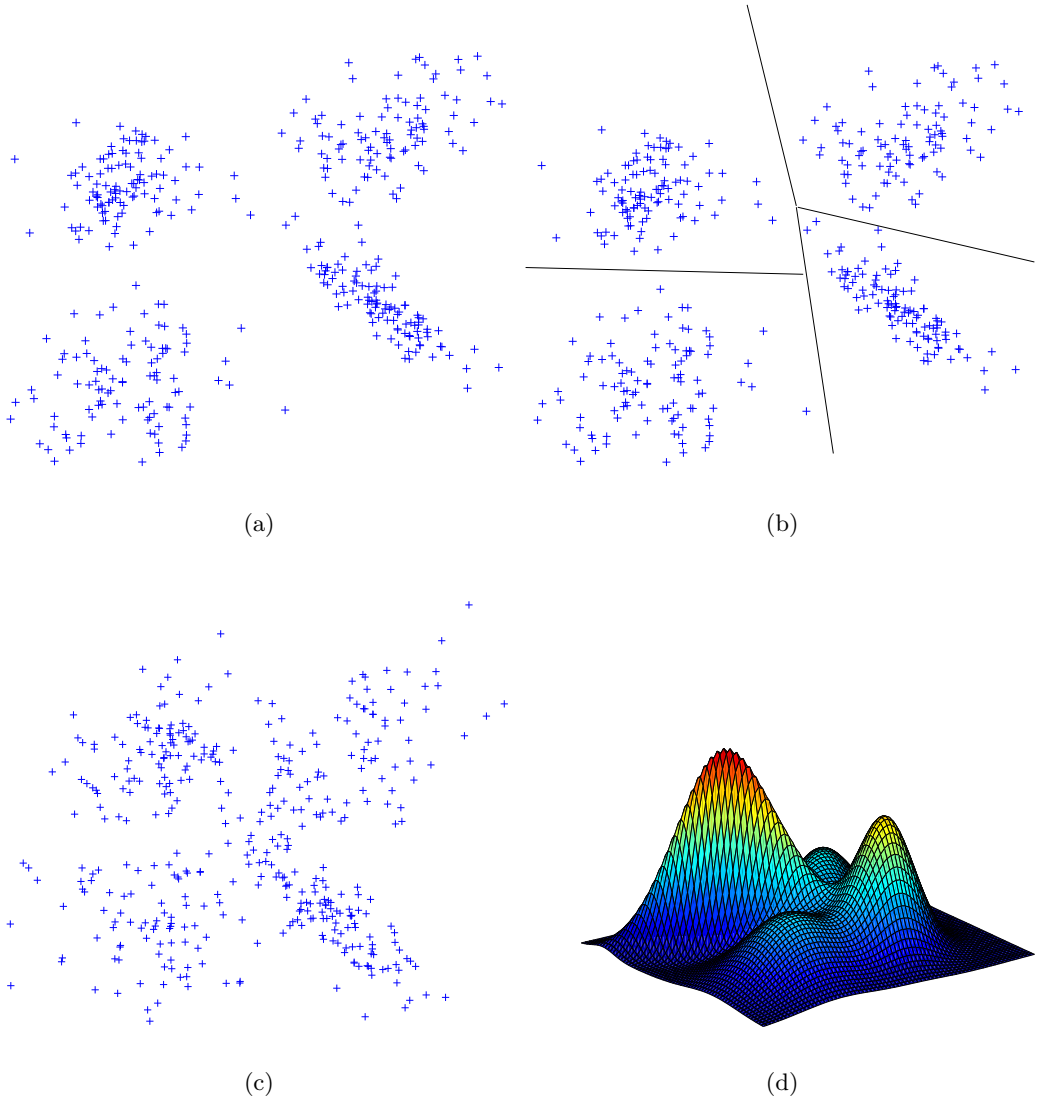
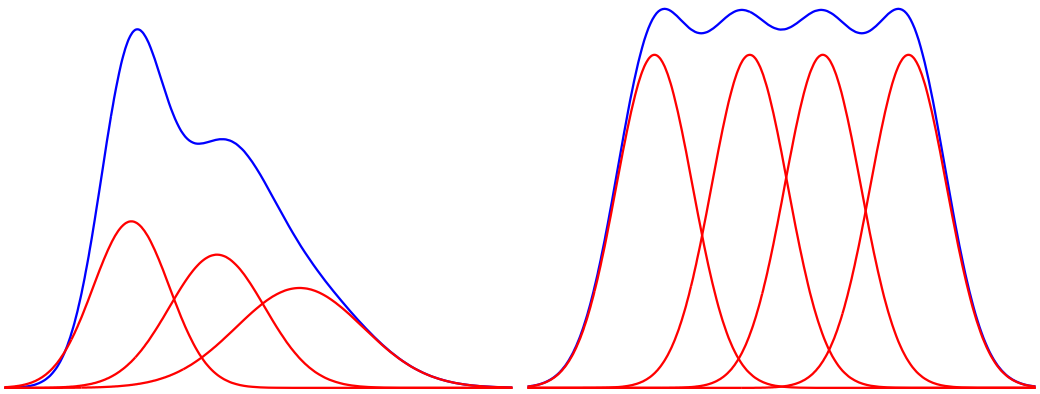


Figure 3.3: (a) A scatter diagram for a well separated clusters. (b) Hard clustering linear boundaries for the data set shown in (a). (c) A scatter diagram for a an interfered clusters. (d) A mesh for a MoG trained on the data set shown in (c).



(a) (b)

Figure 3.4: (a) A triangle density model approximated by a MoG. (b) A uniform density model approximated by a MoG. The red lines show the component density functions of the mixtures and the blue line show the Mixture density functions.

CHAPTER 4

Application: data classification

4.1 Introduction

This chapter describes our experiments for testing the performance of the proposed greedy VD-MPPCA training scheme, on data classification problems. This section is followed by two main parts, that describe two sets of experiments: handwritten digit classification and ionosphere signal classification. Each part starts by describing the data set and the results obtained so far using different classification methods. Each part then reports and discusses the experiments result. Section 4.4 gives concluding remarks. The experiments appear in a chronological order. Therefore, the earlier experiments conclusions result in some modification for the later experiments setting. To compare the performance of the greedy VD-MPPCA with its predecessors, we have designed six experiments, summarized in Table 4.1 and explained below.

I. Training an MPPCA model with a fixed number of subspaces of fixed dimensionality, using randomly initialized EM (Tipping & Bishop (1999*a*) MPPCA's EM).

II. Training an MPPCA model with a fixed number of subspaces of fixed dimensionality, using our greedy hard clustering algorithm only, Algorithm 3; i.e. no EM fitting.

III. Training an MPPCA model with a fixed number of subspaces of fixed dimen-

sionality.

IV Training an MPPCA model with a variable number of subspaces of fixed dimensionality.

V Training a model with a fixed number of subspaces of variable dimensionality.

VI Training a model with a variable number of subspaces of variable dimensionality.

Experiment I is based on the standard EM for MPPCA, typically as described by Tipping & Bishop (1999*a*). In Experiment II we report the performance of our greedy hard clustering method, Algorithm 3. The basic goal of this hard clustering method is the initialization of the greedy EM algorithm. However, if its performance is acceptable it could be used as a stand alone classification method. In addition, reporting its own performance may help in analyzing and comparing the results of the EM based methods. In addition Experiments III through VI are initialized by our greedy hard clustering algorithm. Experiment VI uses our greedy learning scheme as described in Chapter 3. Experiments IV and V use simplified version of our scheme to stop the irrelevant features. For the purpose of comparison, in Experiments V and VI we set α to the average retained variances found by the models generated by Experiments III and IV respectively. All the experiments are repeated five times with differently drawn train and test sets. Experiment I is repeated three times for each pair of sets. The classification is done by assigning each test pattern $\mathbf{y}$ the label of the class i', for which $f_k(\mathbf{y}|\boldsymbol{\theta}_i)$ is highest.

$$i' = \underset{i}{\operatorname{argmax}}\, f_k(\mathbf{y}|\boldsymbol{\theta}_i)$$

where $f_k(\mathbf{y}|\boldsymbol{\theta}_i)$ is the mixture function given by Equation 3.4.1. During the training phase, only patterns of one class are presented to the model generator program, i.e. each class model is built separately.

4.2 Handwritten digit recognition

Handwritten digit recognition is a popular classification problem that is used extensively in testing relative density classification approaches as well as discriminative

Table 4.1: This table summarizes the six experiments. #Sub. and #Dim. are the average number of subspaces and dimensionality per class.

Exp.	Model fitting	#Sub.	#Dim.	Initialization
I	FD-MPPCA EM	fixed	fixed	random
II	hard clustering Algorithm 3	fixed	fixed	
III	FD-MPPCA EM	fixed	fixed	Algorithm 3
IV	greedy FD-MPPCA EM	variable	fixed	Algorithm 3
V	VD-MPPCA EM	fixed	variable	Algorithm 3
VI	greedy VD-MPPCA EM	variable	variable	Algorithm 3

approaches (Hinton et al. 1997). The popularity and the availability of large data sets enable it to stand as a good benchmark for testing and comparing different classification methods. Especially for our problem, there are some publications on handwritten digit classification using mixtures of PCA (Bailing et al. 2001, Hinton et al. 1997, Kim et al. 2002*b*, Tipping & Bishop 1999*a*).

The data set used in our experiments is extracted from the well-known NIST handwritten digit database (Winson & Garris 1992). The original data set consists of 128x128 pixel binary images. In pre-processing, these images are normalized for position, size, slant and stroke width, resulting in 16x16 pixel grey-value images (de Ridder et al. 1996). Furthermore, for the experiments described in our studies, PCA was used on the entire data set to reduce the number of dimensions from 256 to 64. The resulting data set was used to construct training and testing sets. The data set has already been investigated using a number of methods (de Ridder 2001, Winson & Garris 1992). Table 4.2 gives an overview of the results obtained thus far on a training set of 1000 samples per class. 1000 patterns (images) per class (digit) have been used for training and 1000 patterns (images) per class (digit) have been used for testing. For experiments with fixed number of subspaces, each digit is modelled with ten subspaces. For fixed dimensionality experiments each subspace has 10 PCs.

At first, in some of the experiments the EM algorithm did not converge. Therefore, the covariance matrix $\mathbf{C}$ is regularized by adding a small constant value, 0.01, to the

Table 4.2: Results for various classifiers on the NIST data set (de Ridder 2001, Winson & Garris 1992).

Type	Classifier	Error (%)
Bayes plug-in	Nearest mean	15.88
	Linear	9.84
	Quadratic	4.70
Neural network	LeNotre	4.87
	LeNet	3.43
	LeCun	2.32
	1 hidden layer @ 256 units	2.44
	1 hidden layer @ 512 units	1.99
Support Vector Classifier	Polynomial, 5^{th} degree	1.29
	Radial basis, $\sigma = 10$	1.38

parameter σ^2 in each iteration (see Equation 2.5.5).

4.2.1 Results and discussion

Table 4.3 summarizes the first testing results. The error in this table is the average percentage of misclassified patterns in the test set. As an illustration, Figure 4.6 shows some of the means and the PC's found in Experiment VI.

The results of this first set of experiments show that the performance of all methods of initialization and model fitting are nearly equal. Especially the fact that the hard clustering method, Algorithm 3, performs as well as the EM algorithm is curious. To investigate what caused this, we inspect the models found by each experiment. It became then obvious that for some models, problems in estimating σ^2, the noise level, caused poor performance. The regularization value introduced to help the EM algorithm converge, 0.01, seems to have been too small.

The technique, adopted by PPCA, of approximating the noise parameter σ^2 by the average of the minor eigenvalues as given in Equation 2.5.10, gives insight into this problem. Equation 2.5.5 shows that the diagonal components of the model covariance

Table 4.3: Test results, as average error percentage and standard deviation, for the six experiments on NIST handwritten digit data set. In all models, the estimated covariance matrices $\mathbf{C}$ are regularized by adding 0.01 times the identity matrix.

Exp.	Initialization	#Sub.	#Dim	Error	#Sub.	#Dim.
I	random	fixed	fixed	2.66 ± 0.21	10	10
II	Algorithm 3	fixed	fixed	2.66 ± 0.18	10	10
III	Algorithm 3	fixed	fixed	2.67 ± 0.13	10	10
IV	Algorithm 3	variable	fixed	2.60 ± 0.18	5.2	10
V	Algorithm 3	fixed	variable	2.41 ± 0.14	10	10
VI	Algorithm 3	variable	variable	2.60 ± 0.14	7.1	10

matrix $\mathbf{C}$ are dependent on σ^2 in the minor eigenvector directions. This structure makes the covariance matrix very sensitive to the actual value of σ^2. For very small values, $\mathbf{C}$ becomes singular and the whole model becomes undefined. However, even when the matrix is non-singular and σ^2 is very small, the model becomes prone to overfitting. This can be seen by realizing that for small σ^2, some elements of $\mathbf{C}^{-1}$ becomes very large. Now, normally the image elements which are multiplied by the large values in $\mathbf{C}^{-1}$ (see Equation 2.5.6) are very small, as their variances are very low. However, if in the data set, an image occurs which has some noise present in pixel positions which normally have low variance, this noise will be blown up. It will have a large effect on the estimate of the probability of the image and both training (specifically, the E-step) and recognition will suffer.

This is the main reason that makes the clustering-only experiments (Experiment II) give results comparable to the EM-based experiments (Experiments I,III,IV,V and VI), as the noise has less influence on Algorithm 3 than on the EM algorithm. Table 3.1 shows the average σ^2 (over all subspaces) and the recognition error for each class for one of Experiment I. It is obvious from the table that digit "1" has one of the worst results and the lowest value of σ^2. This gives the insight that, σ^2 can be used as a clue for deciding on the optimal number of PCs for the PPCA model in general and most importantly that regularizing σ^2, e.g. by adding a larger regularization

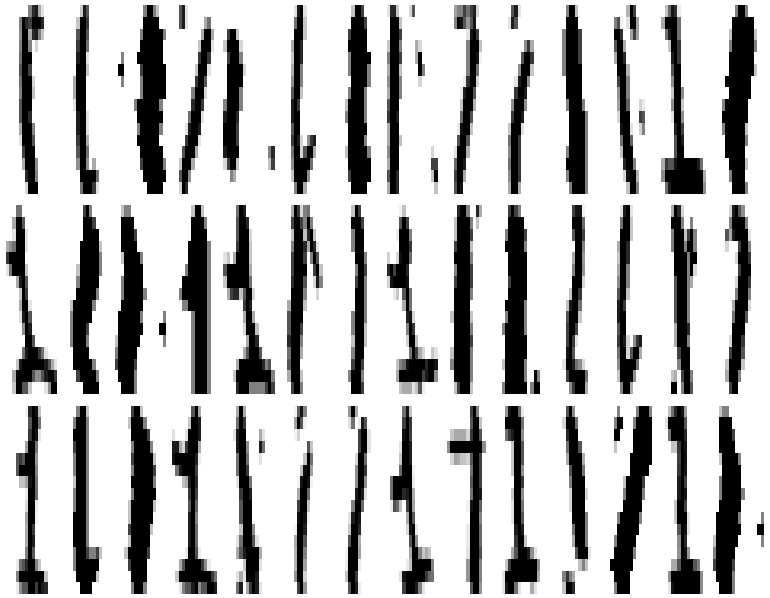


Figure 4.1: Examples for the misclassified digit "1" images in the first set of experiments, In these experiments the estimated covariance matrices $\mathbf{C}$ are regularized by adding 0.01 times the identity matrix. Note how most of the digits have general reasonable shape with some noise presence.

constant, could improve performance. Fig. 4.1 shows some of digit "1" images which have got wrong classification, one can easily note that most of the images have general reasonable shape with noise presence.

To verify this, i.e., the regularization value is small, we re-run all the experiments seven times using different regularization values. In the first run, σ^2 is calculated by averaging the minor eigenvalues, as given in Equation 2.5.10, and adding a fixed regularization constant of 0.05. We then iteratively increased the regularization constant by 0.05 and repeated all experiments. For completeness, the σ^2's estimated by Algorithm 3 (Experiment II) were also regularized in the same way before testing. The results of these experiments, shown in Figure 4.2, show that an optimal regularization value is expected to be around 0.2. Detailed results for the experiments run with a regularization value of 0.2 are shown in Table 4.4. The experiments also show that randomly initialized PPCA now performs quite good compared to the results previously obtained using other classifiers (Table 4.2). It is also obvious that EM is

Table 4.4: Test results for the six experiments using a regularization value of 0.20.

Exp.	Initialization	#Sub.	#Dim.	Error	#Sub.	#Dim.
I	random	fixed	fixed	1.91 ± 0.24	10	10
II	Algorithm 3	fixed	fixed	2.62 ± 0.13	10	10
III	Algorithm 3	fixed	fixed	1.74 ± 0.18	10	10
IV	Algorithm 3	variable	fixed	2.09 ± 0.35	4.5	10
V	Algorithm 3	fixed	variable	1.72 ± 0.11	10	10
IV	Algorithm 3	variable	variable	1.85 ± 0.12	7.2	10

Table 4.5: This table summarizes the result for suboptimal α_{Opt} value, 0.77, result. The second row shows the average number of subspaces for each class. The third row shows the average subspace dimensionality for each class. The fourth row shows the error for each class. The last column shows the average for all classes.

Class	0	1	2	3	4	5	6	7	8	9	Avg.
#Sub.	11	5	10	8	8	9	11	8	11	10	9
#Dim.	9	4	10	12	9	11	8	7	10	8	9
Error %	1.0	1.4	1.4	2.5	1.8	1.9	0.9	1.5	2.3	2.4	1.68 ± 0.17

better than our hard clustering method, Algorithm 3 alone. Using Algorithm 3 as an initialization of the EM algorithm improves results somewhat. However, the most interesting result is the fact that the number of subspaces found in experiment IV, is significantly reduced at only a small increase in test error. The greedy algorithm, Algorithm 2 found 5.5 subspaces for each class on average, where the models with fixed number of subspaces used 10 for each class; it gave a test error of 2.07% on average, versus 1.91% on average for the standard MPPCA algorithm.

To have insight into the relation between the retained variance values, α, subspace dimensionality and classification performance, we re-run experiment VI for 15 different α values. This set of experiments are repeated 5 times with differently drawn train and test sets. Figure 4.3 shows the errors and the average dimensionality for all classes for different α.

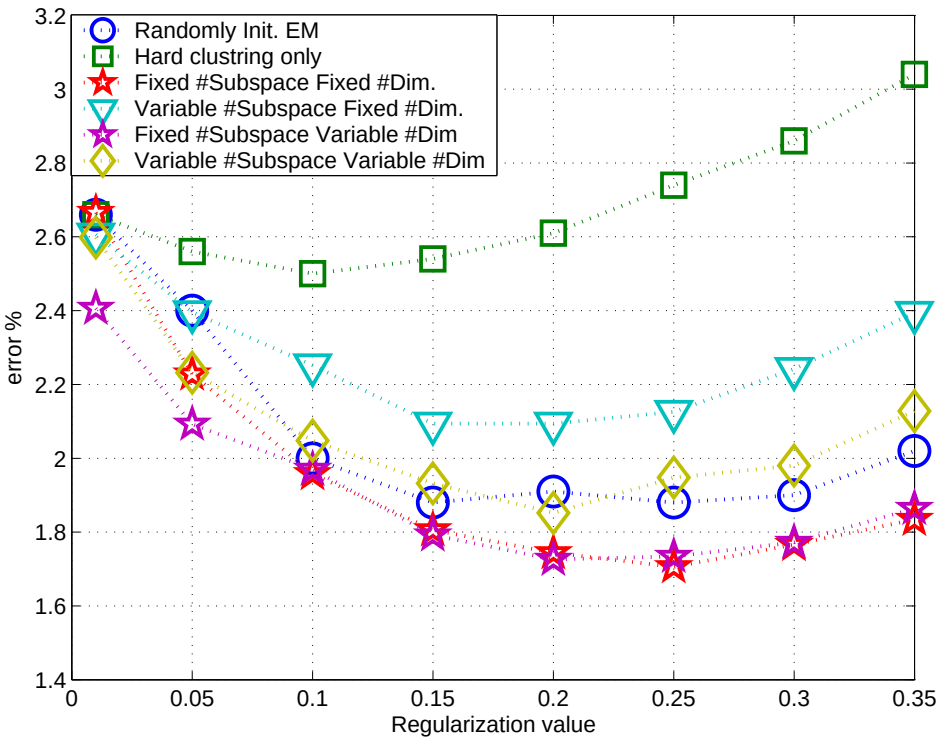


Figure 4.2: Error (% of the test set classified incorrectly) as a function of the regularization value, for experiments I-VI.

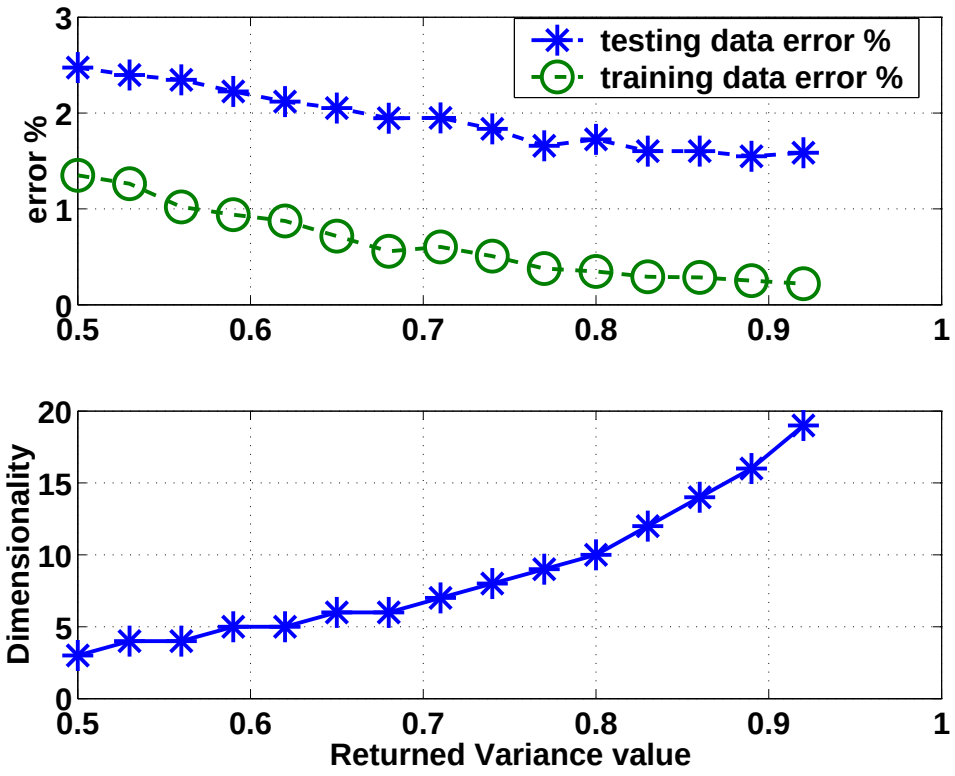


Figure 4.3: (a) Error (average of test and training set classified incorrectly in the 5 experiments) as a function of the retained variance i.e. α. (b) Average subspace dimensionality as a function of α.

Figure 4.3 reflects the important aspect that there exists a suboptimal α (henceforth α_{opt}), here at approximately 0.77. For α values less than α_{opt}, classification performance is proportional to subspace dimensionality. What is of real interest, is that the performance is approximately the same for α values greater than α_{opt} for both training and testing data. In fact there is a very slight improvement that can hardly justify the increase in dimensionality reflected in Fig. 4.3b.

Classes 1, 4 and 7 have the least number of subspaces: 5, 8 and 8, respectively (Table 4.5). On the other hand, classes 0, 6 and 8 have the maximum number of subspaces, 11 subspaces for each. This is impressive, as the latter have more curvature in their shapes and their input space is more full of structure, i.e. more nonlinearity, while the former are semi-straight lines, i.e. less nonlinearity. It shows that the algorithm is not ad hoc in model selection Table 4.5 shows the results per class for the suboptimal α_{opt} of 0.77.

4.3 Ionosphere signal classification

This data set was obtained from the UCI repository, donated by V. Sigillito from the applied Physics Laboratory in Johns Hopkins University (Blake & Merz 1998). The data set consists of two classes of signals, "good" and "bad". Each signal instance has 34 attributes. In accordance with previous experiments, of which results are reported in Table 4.6, we have used 200 instances for training, 100 for each class. The testing set consists of 123 "good" and 24 "bad" instances. The previous results show that "good" signal recognition is much better than the "bad" one.

As before, for Experiments V and VI, we set α to the variance retained by experiments III and IV, respectively. We repeat the experiments 8 times for the same regularization values set used in previous test. We eliminate Experiments I and II. We fix the dimensionality to 5 in the fixed dimensionality experiments and the number of subspaces fixed to 5 in the fixed number of subspaces experiments. We run the same second experiment to see the performance of the model for α values, for which the result is depicted in Figure. 4.5.

Table 4.6: Results for various classifiers on the ionosphere data set.

Classifier	Error %
Linear perceptron	9.3
Nonlinear perceptron	8.0
Backpropagation ANN	4
Nearest neighbor	7.9
Ross Quinlan's C4	6
IB3 (Aha & Kibler IJCAI-1989)	3.3

4.3.1 Results and discussion

Figure 4.4 and Figure 4.5 summarize the results for the given set of experiments graphically. Since the model has its best average performance when the regularization value is 0.1, Table 4.7 gives a detailed result at this regularization value. The figures show that the performance in the 4 experiments is nearly same except for the small regularization values. The performance of experiment VI is better. In Experiment VI the model chooses to model class "good" with an average of 3 subspaces of 3 dimensions each. As these values are better for modelling the "good" class it helps the model to be less vulnerable to noise than in the other experiments where the number of subspaces and dimensionality are higher.

The "bad" class is not as well structured as the "good" class. The model reflects this by choosing 6 subspaces of 8 dimensions each. Estimating 6 subspaces with average dimensionality 8 using only 100 patterns in the training data set produce a model that suffers from the curse of dimensionality problem.

Fig 4.5 shows the performance is almost the same, when α has a value which is greater than or equal to 0.65. In fact this is needed by class "bad" as lower value is quite enough for class "good".

4.4 Conclusions

The results of this classification experiments show that the described model has good performance, when compare with the previously attained results for the given data

Table 4.7: Test results, as average error in % and standard deviation, for the four experiments on the ionosphere data set. #Sub. and #Dim. are the average number of subspaces and dimensionality per class. For all models, the estimated covariance matrices **C** are regularized by a value of 0.1.

Exp.	#Sub.	#Dim.	Error %	#Sub.		#Dim.	
				"good"	"bad"	"good"	"bad"
III	fixed	fixed	3.3 ± 1.1	5	5	5	5
IV	variable	fixed	2.9 ± 0.7	5	4	5	5
V	fixed	variable	3.4 ± 0.8	5	5	3	8
VI	variable	variable	3.4 ± 1.4	3	6	3	8

sets.

The PPCA model's assumption (e.g., equal noise variance in all directions) sometimes cause training problems or poor final performance. To remedy this, simple regularization was shown to improve results considerably. The experiments show that a better regularization value can be found by validation methods. While the main theme of this thesis is making MPPCA training autonomous, this result introduce a new important parameter. Another method (beside validation, which is slow) is needed for setting the regularization value autonomously.

The experiments show that the fixed retained variance value α is a suitable guidance for the process of searching for a suboptimal subspace dimensionality. As α is inversely proportional to the noise term σ^2 and we regularize the model by adding a small constant to σ^2 this suggest that these two parameters should be studied together and it seems that their suboptimal values need to be optimized in a single process. Figure 4.2 and Figure 4.4 show that the models with variable dimensionality have better performance for low regularization values. As the search for α_{opt} and regularization value in our experiments is not exhaustive, more closer to optimality values could be found. The experiments show that α_{opt} can be reached by validation method. Optimizing α and regularization value together by validation methods only increases the complexity quadratically, which may not be feasible for high dimensional data

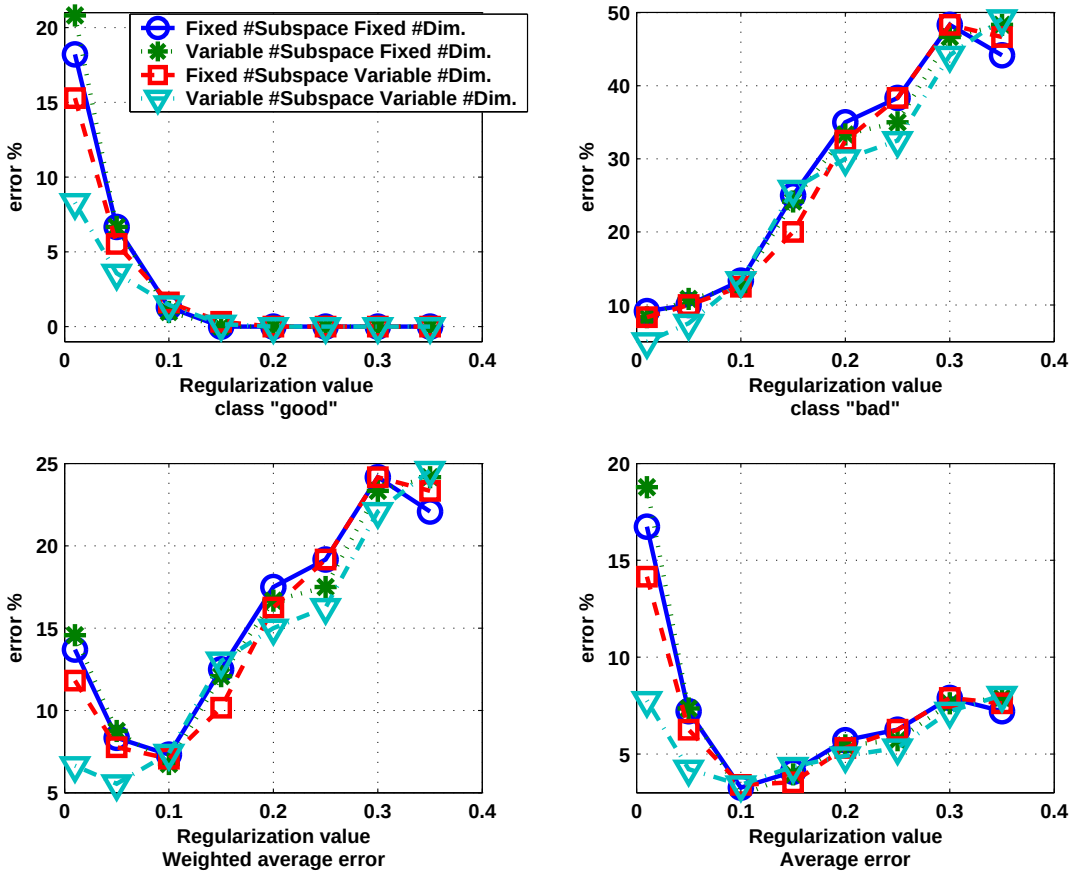
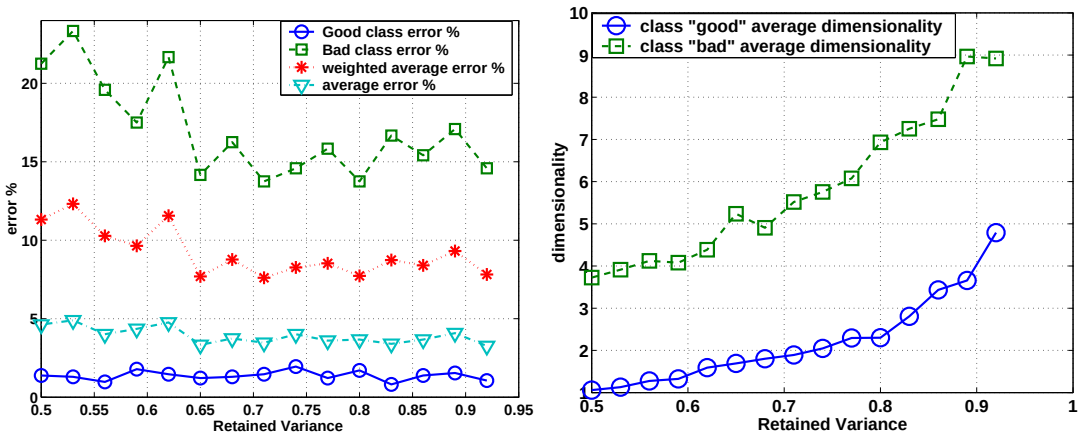


Figure 4.4: Error (% of the test set classified incorrectly) as a function of the regularization value.

sets. This is a subject of further theoretical as well as experimental research studies.

The experiments also show that increasing the number of subspaces iteratively through the greedy training process, is effective in searching for the suboptimal number of subspaces. The process of initializing the greedy EM algorithm by Algorithm 3 (the greedy hard clustering algorithm) shows good results in making The greedy EM less vulnerable to the problem of convergence towards singular estimate at the boundary of the parameter space or stuck in local minimas.



(a) (b)

Figure 4.5: (a) Error (% of the test set classified incorrectly) as a function of α values. (b) Average dimensionality as a function of the retained variance (α).

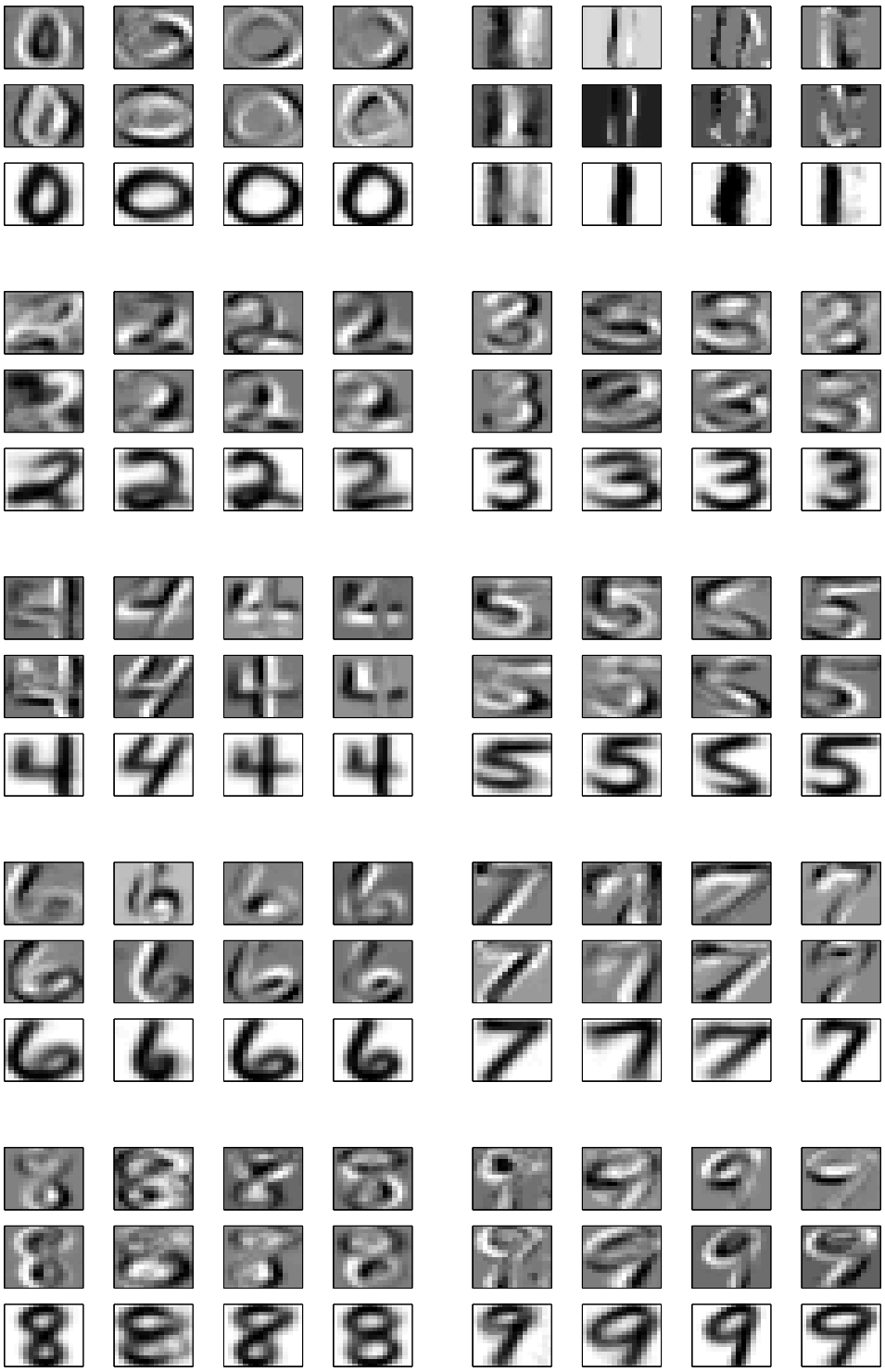


Figure 4.6: Some of subspace origins and the first two PC's found in one of Experiment VI on NIST digit data

CHAPTER 5

Application: texture image segmentation

5.1 Overview

An image (or a patch of an image) is a function $I(x, y)$ on a rectangular grid of pixel positions. For application of pattern recognition techniques, the function values are usually stored in a $(x \times y)$-dimensional vector, by lexicographic ordering of the pixel elements. Treating entire image in this way is computationally infeasible. Therefore, images are normally represented as distribution of d-dimensional vectors ($d = x \times y$). The high dimensional space in which these vectors reside will never be entirely filled. The set of images that make sense to human observer is only very small subset of all possible image function values. Image neighboring pixels are normally highly correlated, and coherent regions in images (e.g., textures) can be better described by individual local models in the high dimensional space. In other words a representation of images (or image patches) as vectors in a high dimensional space contains far more parameters than needed. Lu et al. (1998) had shown that transformed versions of an image patch all lie on an m-dimensional manifold in the d-dimensional space spanned by all pixel values, where m is number of degrees of freedom represented in the transformation. Although this manifold may be intrinsically low-dimensional, its likely to be nonlinear and lie folded up in the d-dimensional space. A good representation would therefore be one which describe this manifold using a small number of parameters, thereby avoiding the estimation problems in high dimensional spaces. Nonlinear

subspace model are theoretically plausible for modelling this manifold. However as we illustrate in Chapter 1, using mixture of linear local models (e.g., MPPCA, mixtures of FA, mixtures of ICA) to approximate nonlinear manifolds is currently showing better results and many recent publications reflect this trend (de Ridder et al. 2000*b*, Moghaddam 2002, Hinton et al. 1997, Kim et al. 2002*a*, Kambhatla 1995, Leen 1997, Sung & Poggio 1998).

Texture image can often be described in a much lower number of parameters than the pixels in the original image, due to the large redundancy in ordinary images and the fact that neighboring pixels are highly correlated. Therefore, it can more naturally be represented by subspaces, with points in the subspace corresponding to slightly translated, rotated, scaled etc. version of the same image. The subspace then becomes an invariant description of an image (or image patch). Moreover, as image normally contains more than one texture, its naturally to use a mixture of subspaces to model it. A leading work in this concern is Kohonen et al. (1997*b*)'s ASSOM (The Adaptive-Subspace Self-Organizing Map), which is an extension of the standard SOM, that uses subspaces in each node instead of weights, that just represent a point in the feature space. This representation enables each node to represent a certain texture in a translation-, rotation- or scale-invariant manner. The result of ASSOM is encouraging. However, as ASSOM need extremely long time for training (Kohonen et al. 1997*a*), researchers' interest is shifted to faster subspace mixture model such as MPCA (de Ridder et al. 2000*a*,*b*). The experiments reported here is a continuation for de Ridder et al. (2000*b*) experiments. de Ridder et al. (2000*b*) have used a two 2-dimensional MPPCA model to model a a 2-texture images. As a result of this continuation, we didn't perform the package of experiments described in Chapter 4 and instead we first repeat experiment de Ridder et al. (2000*a*) with a variable dimensionality model (VD-MPPCA) with the number of subspaces fixed to two. We then apply the greedy VD-MPPCA to allow the model represents each image by more than one subspace.

5.2 Texture image data set

Natural texture images from the Brodatz album are artificially combined (using a cross-shaped mask) to create 2-texture images and scaled to a range [0 1]. The images are shown in the first row of Figure 5.1. The training data consists of 1500 image batches extracted form the combined images, where the length of the batch side is 20 pixels. The entire data set is pre-processed by PCA to remove noise directions. 70 dimensions are left from the original 400 dimensions. The model covariance matrices $\mathbf{C}$ are regularized by adding a value of 0.01 to σ^2. The segmentation is done by assigning each central pixel of an image batch the label of that subspace j for which $p_j(\mathbf{y}|\theta_j)$ is highest. $p_j(\mathbf{y}|\theta_j)$ is the component density that represents subspace j in the mixture function, given by Equation 3.4.1 . The performance of both ICA and PCA mixtures on the same images is reported in de Ridder et al. (2000*b*), where each model is containing two 2-dimensional subspaces.

5.3 Results and discussion

Figure 5.1 shows the resulting segmentations, found by two subspace VD-MPPCA model. It is clear that the result is poor, even for higher α values; i.e., higher dimensionality, there is no obvious improvement. Some parts of the segmented images show that the model subspaces tend to model general image features, such as light intensity, rather than modelling the textures exist in the images, e.g., see Image 4. Moreover, many textures seem to have more than one mode, and therefore need to be modelled by more than one subspace. From these we infer that, restricting the number of subspaces to two is the main reason behind the poor performance. A model with higher number of subspaces may satisfy the needs for more subspaces of some textures and can have some subspaces localized to the common image features. To test this, we re-run the experiments with the number of subspaces 3 through 10. As the pervious set of experiments show no significance improvement by increasing the retained variance ratio (α), in these experiments, we set α to 0.15.

Figure 5.2 shows the results of the second set of experiments. The third row shows the result of the first experiment in which the number of subspaces is 3, the

fourth row shows the result of 4 subspaces, and then each subsequent row has one additional subspace. It is clear that having more than 2 subspaces, the model succeeds in segmenting some images, by having some subspaces localized to one texture in the image. Image 2 shows interesting result as the white subspace marked the boundary area clearly. In image 5, the red subspace localized to the dark area in the outer texture, it is clear that this area has different mode than the other part of the texture and need to be modelled by a separate subspace. For all the ten different subspace settings, image 6, the most difficult image, shows almost the same bad segmentation result.

To see the automatic performance of the model Figure 5.4 shows the best segmentation result based on the highest likelihood criterion. In our data classification experiments, we base the number of subspaces selection entirely on likelihood change criterion.; i.e we stop increasing the number of subspaces when there is no significance change in the likelihood. However, as our primary investigation showed that the likelihood is not properly concave as a function of subspaces k for this data set, see Figure 5.3, we decide to generate 9 models first and to select the one with the highest likelihood.

It is clear the is an over segmentation not a proper segmentation. However by some post processing steps one may produce a typical segmentation from this over segmented images. A general scheme for these post processing steps might be: [1] remove the subspaces that spreads all over the images and assign its points to some neighboring subspaces. [2] combine the subspaces that seam to have the same locality, e.g., by using some neighborhood measurement.

5.4 Conclusions

In this chapter, we apply the greedy VD-MPPCA model to the texture segmentation problem. It is clear form the experiments that increasing the number of subspaces beyond the number of textures in the image, is more efficient than increasing subspace dimensionality. The manifold of one texture seems to be local and low dimensional. However, there is no reason to consider it globally linear, i.e., we may need more

than one subspace to model one texture. Moreover, the common image features, like illumination, and the boundary pixels also play great role in pushing one-subspace-for-one-texture model to get suck in some local maximas. These are the main reasons behind one-subspace-for-one-texture poor performance. However, the final result now is over segmentation that need to be reduced to the proper segmentation. The question of reducing this over segmentation into proper segmentation is a difficult and interesting question that needs more studies.

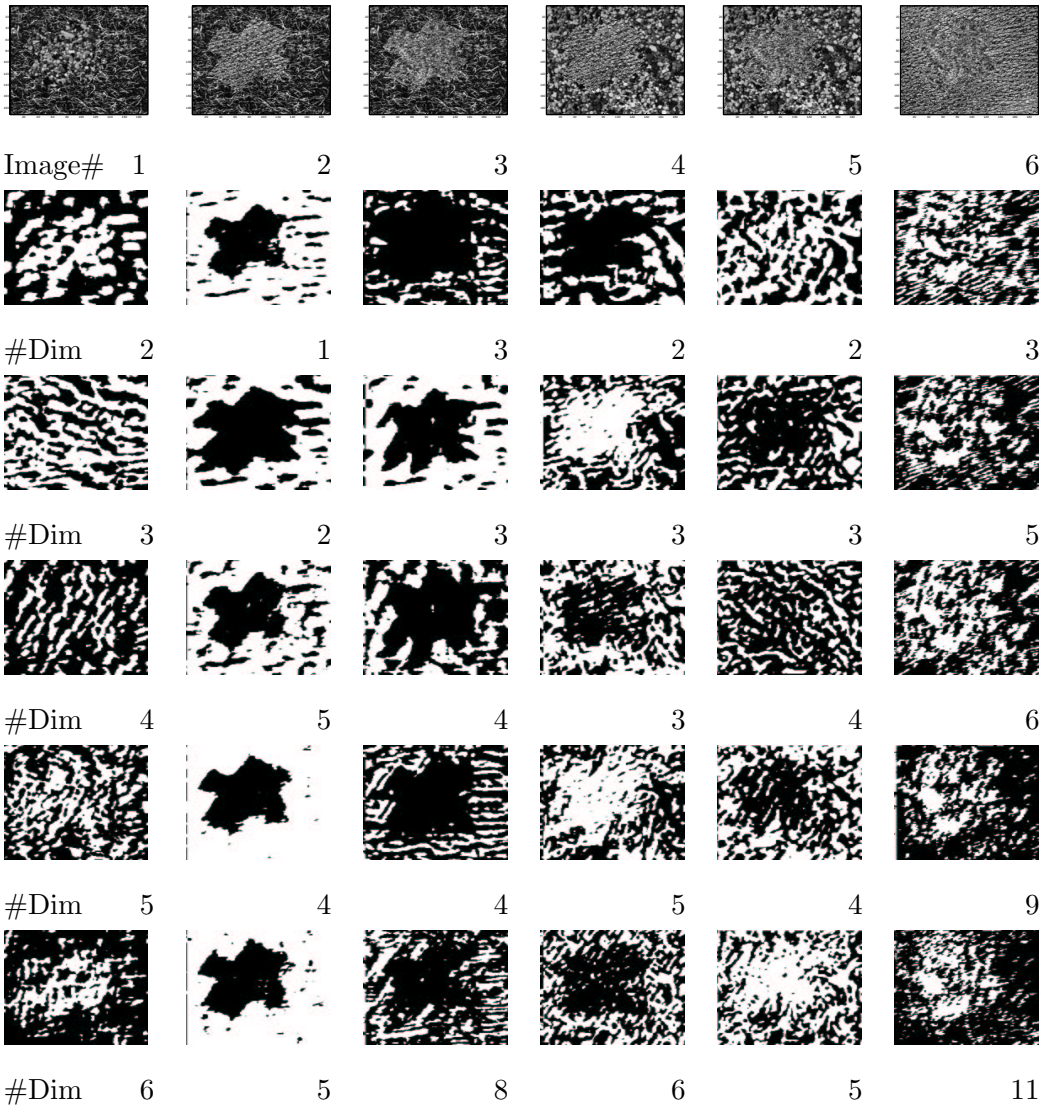


Figure 5.1: Segmentation results for VD-MPPCA model where the number of subspaces is two in all experiments. Each row shows a different experiment with different α values in the following order: 0.15 , 0.20 , 0.25 ,0.30,0.35 and 0.40. The average dimensionality for the subspaces is shown under each image.

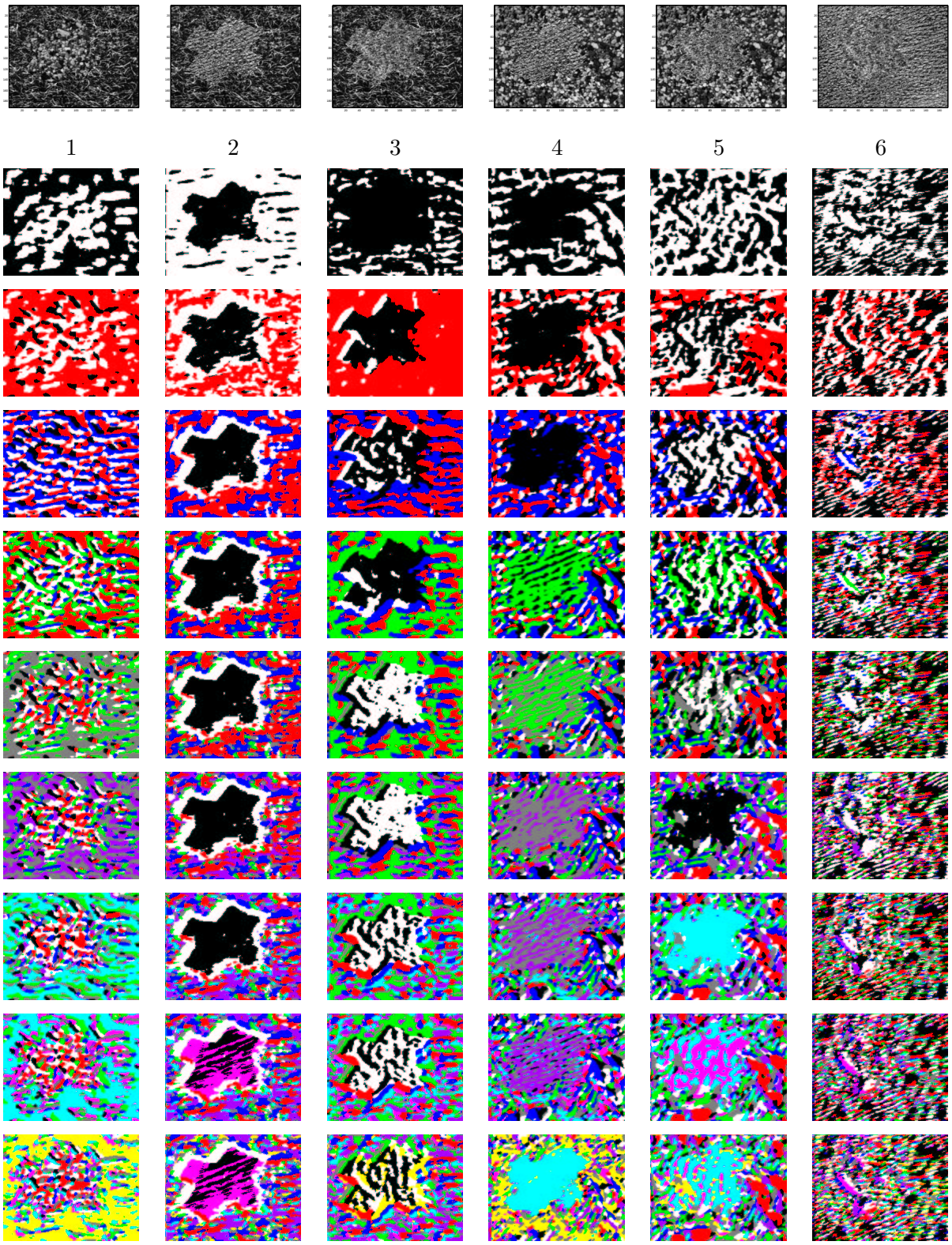


Figure 5.2: Segmentation results of two subspaces through ten subspaces.

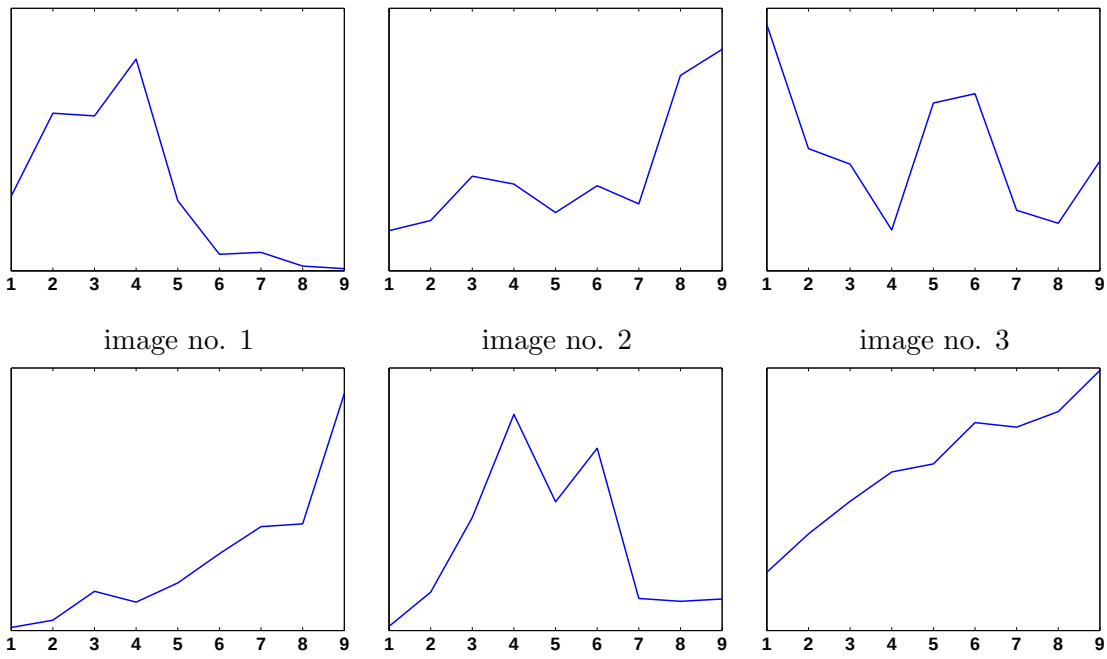


Figure 5.3: Likelihood as a function of number of subspaces

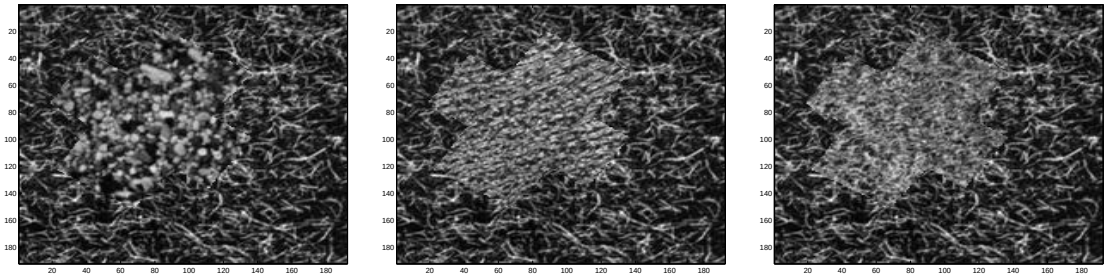
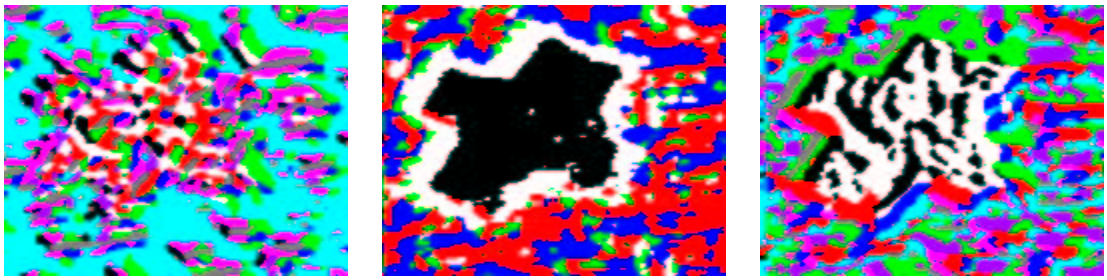


image no. 1 | image no. 2 | image no. 3



Subspaces 9 | Subspaces 6 | Subspaces 9

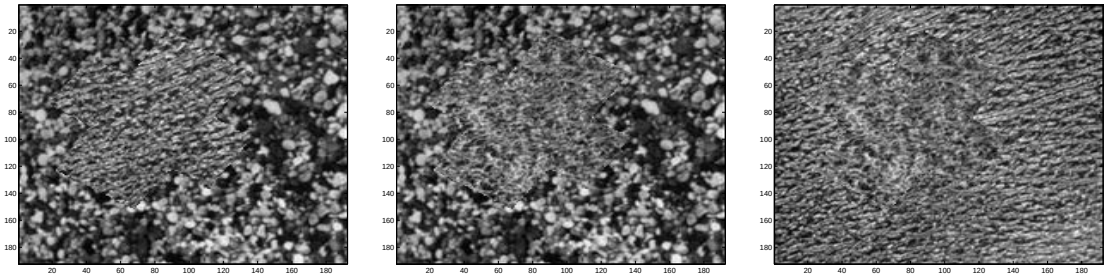
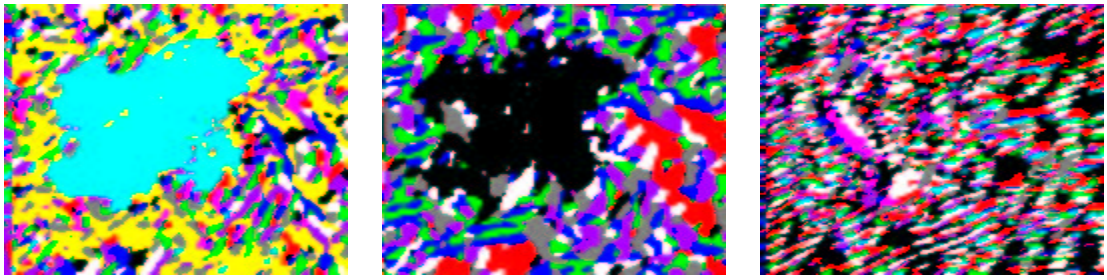


image no. 4 | image no. 5 | image no. 6



Subspaces 10 | Subspaces 7 | Subspaces 9

Figure 5.4: The best number of subspace segmentation results based on maiximum likelihood criterion. The number of subspace is indicated under each segmented image.

CHAPTER 6

Conclusions and future work

MPPCA is an interesting alternative for both PCA and MoG, as it alleviates the PCA global linearity problem and the MoG needs for large number of parameters. This may be the main reason behind the recent research done to enhance MPPCA and extend its applicability. MPCA is the predecessor of MPPCA. MPCA partition the training data into k hard clusters and find principal subspace in each cluster. Both MPCA and MPPCA as described in the literature need the parameters k (number of subspaces) and m (subspace dimensionality) to be set beforehand. The aim of this thesis is the automation of the MPPCA training i.e., finding k and m autonomously. To fulfill this goal, we describe a complete greedy training scheme which combines solutions for automating the MPPCA training. As the new model has variable dimensionality in the local subspaces, we give it the acronym VD-MPPCA.

Finding the optimal number of components in a mixture is a very difficult problem, which is not completely solved yet (McLachlan & Peel 2000). As this is the situation from a purely theoretical point of view, we adopt a simple yet effective and appealing strategy for finding a suboptimal number of components. We set a goal of building a mixture model that adequately represents almost all the training data points. To fulfil this goal, we design a greedy scheme that increases the number of components in the mixture iteratively. Each new component is found by a greedy hard clustering method from a fraction of the training data points, which has the lowest probability

in the current mixture. Finding a new component in the low density regions, increases the possibility of the EM convergence towards singular estimate at the boundary of the parameter space (Figueiredo & Jain 2002). However, our greedy hard clustering initialization method cater for this problem by finding the new component in the most dense region in the given training data fraction.

To find a principal subspace, the user faces two choices: [1] to choose the principal subspace dimensionality and [2] to choose the principal subspace variability. The majority of MPCA and MPPCA described in the literature follow the first choice (Bailing et al. 2001, Dony & Haykin 1997, Hinton et al. 1997, Kambhatla & Leen 1997, Sung & Poggio 1998). We highlight that as the model shifted from geometrically based to be probabilistically based with Tipping & Bishop (1999*b*), Roweis (1997), Moghaddam & Pentland (1997) extensions, the model is expected to be more effective and autonomous by shifting also to the second choice. Accordingly, in our schema we use a fixed retained variance ratio α, for which the scheme chooses the dimensionality that retains this fixed ratio in each local subspace. One may argue that, we still need to specify the global parameter α beforehand. The simple solution for this problem, is to use validation methods (Duda et al. 2000) for finding a suboptimal value α_{opt} for α. What worth mentioning here, is that, the retained variance approach has the advantage of finding subspaces with different dimensionalities and consistent data variability inside and out side the principal subspace.

The main theme of this thesis is making the MPPCA training autonomous. While the greedy search for a suboptimal number of subspaces and the retained variance technique show good results in this concern, unfortunately, the classification experiments show that there is an important additional parameter, namely the regularization, which has to be set. Generally, a regularization value that helps the model to converge can easily be found. However, the experiments show that for some regularization values, hard clustering method performs similar to soft clustering by EM-based methods. This simply means that there is no need for soft fitting by a mixture model. The experiments show that by validation method a better regularization value could be found, for which EM-based methods outperform hard clustering.

The experiments show that the fixed retained variance value α is a reasonable

guidance for the process of searching for a suboptimal subspace dimensionality. As α is inversely proportional to the noise term σ^2 and we regularize the model by adding a small constant to σ^2 this suggest that these two parameters should be studied together and it seems that their suboptimal values need to be optimized in a single process. As the search for α_{opt} and regularization value in our studies is not exhaustive, more closer to optimality values could be found. Optimizing α and regularization value together by validation methods only increases the complexity quadratically, which may not be feasible for high dimensional data sets. An important question is how to find faster method for optimizing these two parameters.

We test the model on data classification. In these experiments we have used two well known data sets: [1] NIST handwritten digit and (Winson & Garris 1992) and [2] UCI ionosphere data set (Blake & Merz 1998). The results of these experiments show that the model has good performance in classification, when compared to the previously attained classification results for these data sets using other (often dedicated) classifiers. k and m values found autonomously by the model for each class in the data sets look consistent. For instance, the model finds higher numbers for k and m for "bad" ionosphere signals with respect to the "good" ionosphere signals. Similar result also appears in the digit classification experiments. For instance, digit "1" k and m values found by the model are the minimum among the digit classes, which is very consistent when considering handwriting styles for different digits.

In the second classification problem (ionosphere data set) which is a two class classification problem, the "bad" signal class has much higher error rates. It seems that exploiting the natural rejection capability of the density models may leverage the performance in such cases. In other words, it may be more efficient to train an MPPCA model for the "good" signals only and to equip this one model with a probabilistic threshold. If a test pattern has a probabilistic membership lower than the threshold it should be rejected. In fact this is one of major benefits of the probabilistic model. However, we think there is a deficiency in investigating this property with MPPCA and therefore it is an important future work.

In the second set of experiments, we test our scheme on segmenting natural texture images from the Brodatz album (Musa et al. 2003). While the result from a pattern

recognition point of view may be interesting, it is an over segmentation from an image processing point of view. As the model finds more than one subspace for one texture. In addition, some subspaces are spread all over the image i.e., not localized to one texture. We think this occurs mainly because the natural texture manifolds are nonlinear. A proper segmentation can be found by applying post processing method on these over-segmented images. This post processing method need to remove the subspaces that spread all over the image first and to combine the subspaces that represent the same texture by using some neighborhood criterion.

One of the MPPCA and MPCA main goals is the approximation of the nonlinear manifold. Local linear embedding (LLE) is a recent proposal for fining a global nonlinear low dimensional projection space (Roweis & Saul 2000, Saul & Roweis 2003). The most important properties of LLE is that it does not contain local minima . However, LLE neither contains clustering nor a probabilistic model. Therefore, LLE is not a stand alone alternative of MPPCA and combining LLE and MPPCA in one setting may be efficient in finding a new powerful method. An interesting future work, may be to investigate using LLE in the post processing step needed for reducing the texture over-segmentation into proper segmentation.

6.1 Training set size dimensionality and number of subspaces

An important issue in statistical pattern recognition is the relation between the training set size, dimensionality and the number of parameters to be found (Krishnaiah & Kanal 1982). Finiteness of the training set size is a factual constraint that cannot be overcome in many practical situations. As a result in modelling a care should be taken as many methods need relatively large training set while some methods show relative tolerance to this problem (Krishnaiah & Kanal 1982, Raudys & Pikelis 1980*a*). VD-MMPCA is a result of successive steps for enhancing the PCA. It seems that some of these enhancements while solving some of PCA major drawbacks; it has the drawback of requiring larger sample size. For instance, an MPCA model requires $\mathcal{O}(k)$ sample size as big as required by conventional PCA. Of course, a soft fitted mixture can tolerate a sample size which is smaller than what is needed for hard fitted mixture.

However the number for both is still higher than what is needed for one global model especially if the mixture contains high number of subspaces care for sample size is crucial.

For FD-MPPCA the modeler can care for the finiteness of the available sample size when choosing k (number of subspace) and m (local dimensionality). Therefore, our greedy VD-MPPCA needs to have some automatic control that prevents it from peaking by having many subspaces with high local dimensions for a relatively small training data set. This problem is very obvious in the model generated for the "bad" ionosphere signals. In the literature there many guidelines for setting some limits for the relation between the training data size and the dimensionality like saying the number of sample per dimensions should be greater than 5 or 10. Many researchers argued that the ratio of sample size to dimensionality should be inversely proportional to the amount of knowledge about the class conditional densities (Kanal & Chandrasekaran 1971). Furthermore, for Gaussian distribution some researchers have designed some tables to determine the optimal dimensionality for a given training data size (Krishnaiah & Kanal 1982, Raudys & Pikelis 1980*a*). Generating or adjusting these tables for PPCA is interesting and worth consideration. Therefore, the greedy VD-MPPCA which search iteratively to find a mixture of Gaussians that fit the given training data can be controlled by this PPCA table to stop increasing the number of subspaces when the table shows that the model is about peaking. Moreover, this control may also reduce the dimensionality when the model is about peaking i.e stopping α control and reduce the model to be FD-MPPCA. These are very acute measurements that need more intensive theoretical and practical studies.

6.2 Complexity issues

Almost all mixture training methods start by obtaining a set of candidate models (usually generated by EM) for a range of values of k (from k_{min} to k_{max}) which is assumed to contain the optimal k. The optimal k is then selected by some model selection criterion. For the described greedy search method the range of models is from k_1 to k_{opt} i.e. the model to be selected is the model with maximum number of

components. Therefore, the greedy search is considered more efficient because it does not generate models with number of components greater than the suboptimal number and this also makes it less vulnerable to peaking when the training set is small.

Starting the number of candidate models by a model which has only one component slows down the training process, especially if the number of subspaces to be found is high. As the greedy hard clustering algorithm can find more than one subspace at a time, the model can be enhanced to start the training process by some k_{min} number of hard clusters. The question is how to find this number k_{min} autonomously? This is a subject for further research works on speeding up the training process.

One iteration in the EM training has the complexity of $\mathcal{O}(kNd^2)$ where k is the number of subspaces, N training size and d is the data dimension. Therefore the whole greedy training process has the complexity $\mathcal{O}(k^2Nd^2)$. The greedy hard clustering needs $\mathcal{O}(\frac{1}{k}N^2d)$ to generate the distance matrix between all $(\frac{1}{k}N^2)$ data points and $\mathcal{O}(N^2logN)$ to sort them to find the smallest radius and $\mathcal{O}(\frac{1}{k}N)$ for nearest mean algorithm, all these can be considered roughly $\mathcal{O}(N^2d)$. Therefore the complexity of the greedy VD-MPPCA can be considered quadratic in the number of components k, data dimension d and the sample size N.

Bibliography

Aitkin, M. & Aitkin, I. (1994), 'Efficient computation of maximum likelihood estimates in mixture distributions with reference to over dispersion and variance components', *Proceedings XVIIth International biometric conference* pp. 123–138.

Bailing, Z., Minyue, F. & Hong, Y. (2001), 'A nonlinear neural network model of mixture of local principal component analysis: application to hahdwritten digit recognition', *Pattern recognition* **34**, 203–214.

Basilevsky, A. (1994), *Statistical factor analysis and related methods*, John Wiley & Sons, Inc.

Bellman, R. (1961), *Adaptive Control Processes*, Princeton University Press.

Bishop, C. M. (1995), *Neural networks for pattern recognition*, Oxford University Press.

Blake, C. & Merz, C. (1998), 'Uci repository of machine learning databases. department of information and computer science', *University of California* .

Cadez, I. V. & Smyth, P. (2000), 'On model selection and concavity for finite mixture models', *Symposium on Information Theory (ISIT)* .

Dasgupta, S. (1999), 'Learning mixtures of gaussians', *Proc. IEEE Symposium on Foundation of Computer Science* .

Day, N. E. (1969), 'Estimating the components of a mixture of two normal distribution', *Biometrica* **32**, 363–474.

DeMers, D. & Cottrell, G. (1993), 'Nonlinear dimensionality reduction', *Advances in Neural Information Processing Systems 5*.

Dempster, A. P., Laird, N. M. & Rubin, D. B. (1977), 'Maximum likelihood from an incomplete data via the em algorithm', *Journal of the Royal Satistical Society* **39**, 1–38.

Devijver, P. A. & Kittler, J. (1982), *Pattern recognition, a statistical approach*, Prentice-Hall.

de Ridder, D., Hoekstra, A. & Duin, R. (1996), 'Feature extraction in shared weights neural networks', *Proceedings of the* 2^{nd} *annual conference of the Advanced School for Computing and Image Processing (ASCI)* pp. 289–294.
URL: *http://www.ph.tn.tudelft.nl/∼dick/publication.html*

de Ridder, D., Kittler, J. & Duin, R. (2000*a*), 'The adaptive subspace map for texture segmentation', *Proceedings of the* 15^{th} *IAPR International Conference on Pattern Recognition* **1**, 216–220.
URL: *http://www.ph.tn.tudelft.nl/∼dick/publication.html*

de Ridder, D., Kittler, J. & Duin, R. (2000*b*), 'Probabilistic pca and ica subspace mixture models for image segmentation', *Proc. 11-th British Machine Vision Conference (BMVC 2000)* pp. 112–121.
URL: *http://www.ph.tn.tudelft.nl/∼dick/publication.html*

de Ridder, D. (2001), Adaptive methods of image processing, PhD thesis, Faculty of Applied Science, Delft University of Technology.
URL: *http://www.ph.tn.tudelft.nl/∼dick/publication.html*

Dony, R. D. & Haykin, S. (1997), 'Image segmentation using a mixture of principal component representation', *IEE Proc. Visual Image Signal Process* **144**, 73–80.

Duda, R. O., Hart, P. E. & Stork, D. G. (2000), *Pattern classification, Second edition*, John Wiley & Sons, Inc.

Figueiredo, M. A. T. & Jain, A. K. (2002), 'Unsupervised learing of finite mixture models', *IEEE transactions on pattern analysis and machine intelligence* **24**, 381–386.

Hartigan, J. A. (1975), *Clustering algorithm*, John Wiley & Sons, Inc.

Hastie, T. (1984), Principal curves and surfaces, PhD thesis, Stanford University.

Haykin, S. (1999), *Neural networks: A comprehensive foundation*, Prentice-Hall.

Hinton, G., Dayan, P. & Revow, M. (1997), 'Modeling the manifolds of images of handwritten digits', *IEEE Transaction on neural networks* **10**, 65–74.

Hotelling, H. (1933), 'Analysis of a complex of statistical variables into principal components', *Journal of Educational Psychology* **24**, 417–441.

Hsieh, W. W. (2001), 'Nonlinear principal component analysis by neural networks', *Tellus* **53A**, 599–615.
URL: *http://www.ocgy.ubc.ca/projects/clim.pred*

Jain, A. K. & Dubes, R. C. (1988), *Algorithms for clustering data*, Prentice Hall.

Jain, A. K., Duin, R. P. W. & Mao, J. (2000), 'Statistical pattern recognition: A review', *IEEE Trans. Pattern Analysis and Machine Intelligence* **22**, 4–37.

Jolliffe, I. T. (2002), *Principal component analysis*, Springer.

Kambhatla, N. & Leen, Y. K. (1997), 'Dimension reduction by local linear principal component analysis', *Neural Computation* **9**, 1493–1516.

Kambhatla, N. (1995), Local models and Gaussian mixture models for statistical data processing, PhD thesis, Oregon Graduate Institute of Science & Technology.

Kanal, L. & Chandrasekaran, B. (1971), 'On dimensionality and sample size in statistical pattern classification', *Pattern recognition* **3**, 225–234.

Karmer, M. A. (1991), 'Nonlinear principal component analysis using autoassociative neural networks', *American Institute of Chemical Engineering Journal* **37**, 233–243.

Kaufman, L. & Rousseeuw, P. J. (1990), *Finding groups in data: An introduction to cluster analysis*, John Wiley & Sons.

Kim, H.-C., Kim, D. & Bang, S. Y. (2002*a*), 'Face recognition using the mixture of eigenfaces method', *Pattern Recognition Letters* **23**, 1549–1558.

Kim, H.-C., Kim, D. & Bang, S. Y. (2002*b*), 'A numeral character recognition using the pca mixture model', *Pattern Recognition Letters* **23**, 103–111.

Kohonen, T., Kaski, S. & Lppalainen, H. (1997*a*), 'The adaptive-subspace self-organizing map', *Neural Computation* **11(2)**, 1321–1344.

Kohonen, T., Kaski, S. & Lppalainen, H. (1997*b*), 'The adaptive-subspace self-organizing map (assom)'.
URL: *http://www.cis.hut.fi/wsom97/progabstracts/48.html*

Krishnaiah, P. R. & Kanal, L. N. (1982), *Handbook of statisics 2: classification, pattern recognition and reduction of dimensionality*, North-Holland.

Leen, T. (1997), 'Image dimension reduction by nonlinear and local linear pca', *Web page* .
URL: *http://www.cse.ogi.edu/ tleen/Research/faces.html*

Lee, J. Q. & Barron, A. R. (2000), 'Mixture density estimation', *Advances in Neural Information Processing Systems* .

Lu, H., Fainman, Y. & Hecht-Nielsen, R. (1998), 'Image manifolds', *Applications of artificial neural networks in image processing III, proceeding SPIE* pp. 52–63.
URL: *http://citeseer.nj.nec.com/Lu98image.html*

McLachlan, G. J. & Basford, K. E. (1988), *Mixture models: Inference and applications to clustering*, Marcel Dekker.

McLachlan, G. J. & Peel, D. (2000), *Finite mixtures models*, John Wiley & Sons, Inc.

Meinicke, P. & Ritter, H. (2001), 'Resolution-based complexity control for gaussian mixture models', *Neural Computation* **13(2)**, 453–475.

Moghaddam, B. & Pentland, A. (1997), 'Probabilistic visual learning for object presentation', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **19(7)**, 696–710.

Moghaddam, B. (2002), 'Principal manifolds and probabilistic subspaces for visual recognition', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24(6)**, 780–788.
URL: *www.merl.com/papers/docs/TR2002-13.pdf*

Musa, M. E. M., Duin, R. P., de Ridder, D. & Atalay, V. (2003), 'Texture segmentation using the mixtures of principal component analyzers', *Eighteenth International Symposium on Computer and Information Sciences (ISCIS XVIII), Antalya, Turkey.*

Musa, M. E. M., Duin, R. P. & de Ridder, D. (2001*a*), 'An enhanced em algorithm for mixture of probabilistic principal component analyzers', *ICANN 2001 workshop on kernel & subspace methods for computer Vision.*

Musa, M. E. M., Duin, R. P. & de Ridder, D. (2001*b*), 'Modelling handwritten digit data using probabilistic principal component analysis', *Pro* 7^{th} *Annual conference of the advance School of Computing and Imaging (ASCI)* pp. 415–421.

Pearson, K. (1901), 'On lines and planes of closest fit to systems of points in space', *London, Edinburgh and Dublin Philosophical Magazine and Journal of Science, Six Series* **2**, 559–572.

Petsche, T., Marchatonio, A., Darken, C., Hanson, S., Kuhn, G. & Santoso, I. (1996), 'A neural network autoassociator for induction motor failure predication', *Advances in Neural Information Processing Systems* **8**, 924–930.

Raudys, S. & Jain, A. (1991), 'Small sample size effects in statistical pattern recognitions for practioners', *IEEE Trans. Pattern Analysis and Machine Intelligence* **13**, 252–264.

Raudys, S. & Pikelis, V. (1980*a*), 'On dimensionality, sample size,classification error,

and complexity of classification algorithm in pattern recognition', *IEEE Transaction pattern analysis and machine intelligence* **23**, 242–252.

Raudys, S. & Pikelis, V. (1980*b*), 'On dimensionality, sample size, classification error, and complexity of classification algorithms in pattern recognition', *IEEE Trans. Pattern Analysis and Machine Intelligence* **2**, 243–251.

Render, R. A. & Walker, H. F. (1984), 'Mixture densities, maximum likelihood and the em algorithm', *SIAM review* **26(2)**, 195 – 239.

Ripley, B. (1996), *Pattern recognition and neural networks*, Cambridge University Press.

Roweis, S. & Saul, L. K. (2000), 'Nonlinear dimensionality reduction by locally linear embedding', *Science* **290**, 2323–2326.
URL: *http://www.gatsby.ucl.ac.uk/∼roweis/publications.html*

Roweis, S. (1997), 'Em algorithm for pca and spca', *Advances in Neural Information Processing Systems 10* .
URL: *http://www.gatsby.ucl.ac.uk/∼roweis/publications.html*

Saul, L. K. & Roweis, S. (2003), 'Think globally, fit locally:unsupervised learning of low dimensional manifolds', *Journal of Machine Learning Research* **4**, 119–155.
URL: *http://www.gatsby.ucl.ac.uk/∼roweis/publications.html*

Schalkoff, R. (1992), *Pattern recognition statistical structural and neural approachs*, John Wiley and Sons, Inc.

Smith, L. (1998), *Linear Algebra*, Springer.

Stanford, D. & Raftery, A. E. (1997), 'Principal curve clustering with noise', *Technical Report 317, Department of Statistics, University of Washington* .
URL: *http://www.stat.washington.edu/raftery*

Sung, K.-K. & Poggio, T. (1998), 'Example-based learning for view-based human face detection', *IEEE Transaction on pattern Analysis and Machine Intelligence*

20(1), 39–51.

URL: *ftp://publication.ai.it.edu/ai-publication/1500-1999/AIM-1521.ps.Z.*

Tipping, M. E. & Bishop, C. M. (1999*a*), 'Mixtures of probabilistic principal component analyzers', *Neural Computation* **11(2)**, 443–482.

Tipping, M. E. & Bishop, C. M. (1999*b*), 'Probabilistic principal component analyzers', *Journal of the royal statistic Society* **b 61**, 611–622.

Titterington, D., Smith, A. & Makov, U. (1985), *Statistical Analysis of finite mixtures distributions*, John Wiley & Sons, Chichester, U.K.

Traven, H. G. C. (1991), 'A neural network approach to statistical pattern classification by 'semiparametric' estimation of probability density function', *IEEE Transaction on neural networks* **(2)3**, 264–280.

Verbeek, J. J., Vlassis, N. & Krose, B. (2003), 'Efficient greedy learning of gaussian mixture models', *Neural computation* **15(2)**, 469–485.

Winson, C. L. & Garris, M. D. (1992), 'Handprinted character database 3', *National Institute of Standards and Technology; Advanced Systems Division* p. 43.

URL: *http://www.nist.gov/srd/nistsd19.htm*

Wolfe, J. H. (1970), 'Pattern clustering by multivariate mixture analysis', *Multivariate behavioral research* **5**, 329–350.

APPENDIX A

Eigenvectors derivation for principal components

Let $\mathbf{y}$ be a d-dimensional random vector, which has the covariance matrix $\mathbf{Cy}$, let $\mathbf{A}$ be $d \times m$ projection matrix and let $\mathbf{x}$ be an m-dimensional random vector defined as the projection of $\mathbf{y}$ by $\mathbf{A}$ into some m-dimensional space.

$$\mathbf{x} = \mathbf{A}^T\mathbf{y}$$

Now the covariance $\mathbf{Cx}$ of $\mathbf{x}$ can be defined as follows

$$\mathbf{Cx} = \mathbf{A}^T\mathbf{CyA}$$

or in detailed forms

$$\text{Variance}(\mathbf{x_i}) = \mathbf{A_i}^T\mathbf{CyA_i}$$
$$\text{Cov}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{A_i}^T\mathbf{CyA_j}$$

$\mathbf{x}$ contains the first m principal components (PCs) of $\mathbf{y}$ if its elements are uncorrelated and whose variances are as large as possible. $\mathbf{x}_i$ (PC_1) is the linear combination $\mathbf{A_1}^T\mathbf{y}$ with the maximum variance. That it maximizes the $\text{Variance}(\mathbf{x_1}) = \mathbf{A_1}^T\mathbf{CyA_1}$. It is clear that $\text{Variance}(\mathbf{x_1}) = \mathbf{A_1}^T\mathbf{CyA_1}$ can be increased by multiplying $\mathbf{A_1}$ by some constant. To eliminate this indeterminacy, it is convenient to restrict attention to coefficient vectors of unit length. We therefore define

- $\mathbf{x_1}$ (PC_1) = the linear combination $\mathbf{A}_1^T\mathbf{y}$ that maximizes $\text{Variance}(\mathbf{A}_1^T\mathbf{y})$ subject to $\mathbf{A}_1^T\mathbf{A}_1 = 1$

- $\mathbf{x_2}$ (PC_2) = the linear combination $\mathbf{A}_2^T\mathbf{y}$ that maximizes Variance($\mathbf{A}_2^T\mathbf{y}$) subject to $\mathbf{A}_2^T\mathbf{A}_2 = 1$ and $\mathrm{Cov}(\mathbf{A}_1^T\mathbf{y}, \mathbf{A}_2^T\mathbf{y}) = 0$

- $\mathbf{x_i}$ (PC_i) = the linear combination $\mathbf{A}_i^T\mathbf{y}$ that maximizes Variance($\mathbf{A}_i^T\mathbf{y}$) subject to $\mathbf{A}_i^T\mathbf{A}_i = 1$ and $\mathrm{Cov}(\mathbf{A}_i^T\mathbf{y}, \mathbf{A}_j^T\mathbf{y}) = 0$ for $j < i$.

Theorem

if $\mathbf{Cy}$ has the eigenvalue-eigenvector pairs (λ_1, $\mathbf{e}_1$),(λ_2, $\mathbf{e}_2$), ...,(λ_m, $\mathbf{e}_m$) where $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_m \geq 0$. then

$$\mathbf{A} = \mathbf{e}_1, ..., \mathbf{e}_m$$

and $\mathbf{Cx}$ is diagonal matrix with its diagonal elements satisfy $\mathbf{Cx}_{ii} = \lambda_i$. If some λ_i, are equal, the choices of the corresponding PCs are not unique.

Proof

As $\mathbf{Cy}$ is a positive definite matrix we can use the following positive definite matrix properties

- Property 1
$$\max_{\mathbf{A}_1 \neq 0} \frac{\mathbf{A}_1^T\mathbf{Cy}\mathbf{A}_1^T}{\mathbf{A}_1^T\mathbf{A}_1^T} = \lambda_1 \quad \text{(attained when } \mathbf{A}_1 = \mathbf{e}_1\text{)}$$

- Property 2
$$\max_{\mathbf{A}_1 \perp \mathbf{e}_1, ... \mathbf{e}_i} \frac{\mathbf{A}_j^T\mathbf{Cy}\mathbf{A}_j^T}{\mathbf{A}_j^T\mathbf{A}_j} = \lambda_{i+1} \quad \text{(attained when } \mathbf{A}_j^T = \mathbf{e}_{i+1}\text{)}$$

a proof for the above formulas can be found in many linear algebra text book e.g., Smith (1998).

Since $\mathbf{e}_1^T\mathbf{e} = 1$ the eigenvector are normalized. thus using Property 1

$$\max_{\mathbf{A}_1 \neq 0} \frac{\mathbf{A}_1^T\mathbf{Cy}\mathbf{A}_1^T}{\mathbf{A}_1^T\mathbf{A}_1} = \lambda_1 = \mathbf{e}_1^T\mathbf{Cy}\mathbf{e}_1^T = \mathrm{Variance}(\mathbf{x_1})$$

Similarly, using Property 2 for the choice $\mathbf{A}_j = \mathbf{e}_{k+1}$ with $\mathbf{e}_{k+1}^T\mathbf{e}_i = 0$ for $i = 1, 2, ..., k$ and $k = 1, 2, ..., m-1$, we get

$$\frac{\mathbf{e}_{k+1}^T\mathbf{Cy}\mathbf{e}_{k+1}}{\mathbf{e}_{k+1}^T\mathbf{e}_{k+1}} = \mathbf{e}_{k+1}^T\mathbf{Cy}\mathbf{e}_{k+1} = \mathrm{Variance}(\mathbf{x}_{k+1})$$

But $\mathbf{e}_{k+1}^T(\mathbf{Cye}_{k+1}) = \lambda_{k+1}\mathbf{e}_{k+1}^T\mathbf{e}_{k+1} = \lambda_{k+1}$ so Variance$(\mathbf{x}_{k+1}) = \lambda_{k+1}$. It remains to show that $\mathbf{e}_i$ perpendicular to $\mathbf{e}_k$ (that is $\mathbf{e}_i^T\mathbf{e}_k = 0, i \neq k$) gives Cov$(\mathbf{x_i}, \mathbf{x_k} = 0)$. Now, the eigenvectors of $\mathbf{Cy}$ are orthogonal if all eigenvalues $\lambda_1, \lambda_2, ..., \lambda_m$ are distinct. If the eigenvalues are not distinct. The eigenvectors corresponding to common eigenvalues may be chosen to be orthogonal. Therefore, for any two eigenvectors $\mathbf{e}_i$ and $\mathbf{e}_k$,$\mathbf{e}_i^T$ $\mathbf{e}_k = 0, i \neq k$. Since $\mathbf{Cye}_k = \lambda_k\mathbf{e}_k$, pre-multiplication by $\mathbf{e}_i^T$ gives

$$\text{Cov}(\mathbf{x}_i, \mathbf{x}_k) = \mathbf{e}_i\mathbf{Cye}_k = \mathbf{e}_i^T\lambda_k\mathbf{e}_k = \lambda_k\mathbf{e}_i^T\mathbf{e}_k = 0$$

for any $i \neq k$, and the proof is complete.

APPENDIX B

Expectation maximization algorithm

B.1 Introduction

The Expectation maximization (EM) algorithm is a parameter estimation method which falls into the general framework of maximum likelihood estimation, and is applied in cases where part of the data can be considered to be incomplete, or "hidden". For instance, in mixture parameters estimation, the information that show to which component in the mixture a data point belongs is unknown. Therefore, the absence of this information makes the mixture training data incomplete. EM is essentially an iterative optimization algorithm which, at least under certain conditions, will converge to parameter values at a local maximum of the likelihood function.

Dempster et al. (1977) defined the EM algorithm, and proved certain properties, in particular that at each iteration the loglikelihood of the observed data is guaranteed to be nondecreasing. That is, if $\mathcal{L}(\mathbf{\Theta})$ is the likelihood of the observed data given parameter values $\mathbf{\Theta}$, and $\mathbf{\Theta}_t$, $\mathbf{\Theta}_{t+1}$ are the parameter values at the t'th and $(t+1)$'th iterations respectively, then $\mathcal{L}(\mathbf{\Theta}_{t+1}) \geq \mathcal{L}(\mathbf{\Theta}_t)$.

B.2 EM definition

The EM algorithm assumes the following problem definition: we have two sample spaces $\mathcal{Z}$ and $\mathcal{Y}$, such that there is a many-to-one mapping $\mathbf{y} = f(\mathbf{z})$ from an obser-

vation $\mathbf{y}$ in $\mathcal{Y}$ to an observation $\mathbf{z}$ in $\mathcal{Z}$. We define

$$\mathcal{Z}(\mathbf{y}) = \{\mathbf{z} : f(\mathbf{z}) = \mathbf{y}\} \tag{B.2.1}$$

$\mathbf{z}$ is the complete data, and $\mathbf{y}$ is the observed data. If the distribution $f(\mathbf{z}|\mathbf{\Theta})$ is well defined then the probability of $\mathbf{y}$ given $\mathbf{\Theta}$ is

$$g(\mathbf{y}|\mathbf{\Theta}) = \int_{\mathcal{Z}(\mathbf{y})} f(\mathbf{z}|\mathbf{\Theta}) d\mathbf{z} \tag{B.2.2}$$

EM attempts to solve the following problem: given a sample from $\mathbf{y}$ is observed, but the corresponding $\mathbf{z}$ are unobserved, or hidden, find the maximum likelihood estimate $vec\Theta$ which maximizes $\mathcal{L}(\mathbf{\Theta}) = \log g(\mathbf{y}|\mathbf{\Theta})$. In general, $\log f(\mathbf{z}|\mathbf{\Theta})$ will have an easily–defined, analytically solvable maximum, but maximization of $\mathcal{L}(\mathbf{\Theta})$ has no analytic solution. EM is an iterative optimization algorithm which defines a sequence of parameter settings through a mapping $\mathbf{\Theta}_t \rightarrow \mathbf{\Theta}_{t+1}$ such that $\mathcal{L}(\mathbf{\Theta}_{t+1}) \geq \mathcal{L}(\mathbf{\Theta}_t)$ with equality holding only at stationary points of $\mathcal{L}(\mathbf{\Theta})$. Thus EM is a hill-climbing algorithm which, at least under certain conditions, will converge to a stationary point of $\mathcal{L}(\mathbf{\Theta})$.

The mapping $\mathbf{\Theta}_t \rightarrow \mathbf{\Theta}_{t+1}$ is defined in two steps:

- The **Estimation** step. Define $\widetilde{p}(\mathbf{z}) = p(\mathbf{z}|\mathbf{y}, \mathbf{\Theta}_t)$. Calculate

$$Q(\mathbf{\Theta}', \mathbf{\Theta}_t) = E[\log f(\mathbf{z}|\mathbf{\Theta}')|\widetilde{p}(\mathbf{z}] = \int \widetilde{p}(\mathbf{z}) \log f(\mathbf{z}|\mathbf{\Theta}') d\mathbf{z} \tag{B.2.3}$$

- The **Maximization** step. Set $\mathbf{\Theta}_{t+1} = argmax_{\mathbf{\Theta}'}, Q(\mathbf{\Theta}', \mathbf{\Theta}_t)$.

The intuition is as follows: if we had the complete data, we would simply estimate $\mathbf{\Theta}'$ to maximize $\log f(\mathbf{z}|\mathbf{\Theta}')$. But with some of the complete data missing we instead maximize the expectation of $\log f(\mathbf{z}|\mathbf{\Theta}')$ given the observed data and the current value of $\mathbf{\Theta}$.

B.3 Proof that the likelihood is nondecreasing at each iteration

A crucial property of the EM algorithm is that the loglikelihood $\mathcal{L}(\mathbf{\Theta}) = \log g(\mathbf{y}|\mathbf{\Theta})$ is non decreasing at each iteration. Formally, if we define the EM mapping as $\mathbf{\Theta}_t \rightarrow \mathbf{\Theta}_{t+1}$ where $\mathbf{\Theta}_{t+1} = \text{argmax}_{\mathbf{\Theta}'} Q(\mathbf{\Theta}', \mathbf{\Theta}_t)$ then $\mathcal{L}(\mathbf{\Theta}_{t+1}) \geq \mathcal{L}(\mathbf{\Theta}_t)$. The proof rests on two results:

- Define $k(\mathbf{z}|\mathbf{y},\mathbf{\Theta})$ to be the posterior likelihood of the complete data given the data $\mathbf{y}$ and the parameters $\mathbf{\Theta}$, so that $k(\mathbf{z}|\mathbf{y},\mathbf{\Theta}) = \frac{f(\mathbf{z}|\mathbf{\Theta})}{g(\mathbf{y}|\mathbf{\Theta})}$). If we define $H(\mathbf{\Theta'},\mathbf{\Theta}) = E[\log k(\mathbf{z}|\mathbf{y},\mathbf{\Theta'})|\widetilde{p}(\mathbf{z})]$, (as before, $\widetilde{p}(\mathbf{z}) = p(\mathbf{z}|\mathbf{y},\mathbf{\Theta})$), then

$$\mathcal{L}(\mathbf{\Theta'}) = Q(\mathbf{\Theta'},\mathbf{\Theta}) - H(\mathbf{\Theta'\Theta}) \tag{B.3.1}$$

- $$\forall \mathbf{\Theta'} \;\; H(\mathbf{\Theta'},\Theta) \leq H(\mathbf{\Theta},\mathbf{\Theta}) \tag{B.3.2}$$

  with equality iff $\log k(\mathbf{z}|\mathbf{y},\mathbf{\Theta'}) = \log k(\mathbf{z}|\mathbf{y},\mathbf{\Theta})$ almost everywhere.

Given B.3.1

$$\mathcal{L}(\mathbf{\Theta}_t) - \mathcal{L}(\mathbf{\Theta}_{t+1}) = \{Q(\mathbf{\Theta}_{t+1},\mathbf{\Theta}_t) - Q(\mathbf{\Theta}_t,\mathbf{\Theta}_t)\} - \{H(\mathbf{\Theta}_{t+1},\mathbf{\Theta}_t) - H(\mathbf{\Theta}_t,\mathbf{\Theta}_t)\} \tag{B.3.3}$$

But $Q(\mathbf{\Theta}_{t+1},\mathbf{\Theta}_t) - Q(\mathbf{\Theta}_t,\mathbf{\Theta}_t) \geq 0$ (by the definition of the M step), and from B.3.2 $H(\mathbf{\Theta}_{t+1},\mathbf{\Theta}_t) - H(\mathbf{\Theta}_t,\mathbf{\Theta}_t))$, so clearly $\mathcal{L}(\mathbf{\Theta}_{t+1}) - \mathcal{L}(\mathbf{\Theta}_t) \geq 0$).

B.4 Proof of equation B.3.1

By the rules of conditional probability,

$$\begin{aligned} k(\mathbf{z}|\mathbf{y},\mathbf{\Theta'}) &= \frac{f(\mathbf{z}|\mathbf{\Theta'})}{g(\mathbf{y}|\mathbf{\Theta'})} && \text{(B.4.1)} \\ k(\mathbf{z}|\mathbf{y},\mathbf{\Theta'}) &= \log f(\mathbf{z}|\mathbf{\Theta'}) - \log g(\mathbf{z}|\mathbf{\Theta'}) && \text{(B.4.2)} \end{aligned}$$

We can now take expectations with respect to $\widetilde{p}(\mathbf{z}) = p(\mathbf{z}|\mathbf{y},\mathbf{\Theta})$:

$$\begin{aligned} E[\log k(\mathbf{z}|\mathbf{y},\mathbf{\Theta'}) \mid \widetilde{p}(\mathbf{z})] &= E[\log f(\mathbf{z}|,\mathbf{\Theta'}) \mid \widetilde{p}(\mathbf{z})] - E[\log g(\mathbf{y}|\mathbf{\Theta'}) \mid \widetilde{p}(\mathbf{z})] \\ &= E[\log f(\mathbf{z}|,\mathbf{\Theta'}) \mid \widetilde{p}(\mathbf{z})] - \log g(\mathbf{y}|\mathbf{\Theta'}) && \text{(B.4.3)} \end{aligned}$$

(Note that $E[\log g(\mathbf{y}|\mathbf{\Theta'}) \mid \widetilde{p}(\mathbf{z})] = \log g(\mathbf{y}|\mathbf{\Theta'})$ as $\log g(\mathbf{y}|\mathbf{\Theta})$ does not depend on $\mathbf{z}$.) So by the definitions of H , Q and $\mathcal{L}$,

$$H(\mathbf{\Theta'},\mathbf{\Theta}) = Q(\mathbf{\Theta'},\mathbf{\Theta}) - \mathcal{L}(\mathbf{\Theta'}) \tag{B.4.4}$$

B.5 Proof of equation B.3.2

One thing to note is that $H(\boldsymbol{\Theta}, \boldsymbol{\Theta}) - H(\boldsymbol{\Theta}', \boldsymbol{\Theta})$ is the Kullback-Liebler distance between $k(\mathbf{z}|\mathbf{y}, \boldsymbol{\Theta})$ and $k(\mathbf{z}|\mathbf{y}, \boldsymbol{\Theta}')$, which is known to be ≥ 0 with equality only if the two distributions are equal. A formal proof is through the following theorem. Let $f(\mathbf{z})$ and $g(\mathbf{z})$ be nonnegative and integrable functions, and S be the region in which $f(\mathbf{z}) \geq 0$. The theorem states that if $\int_S (f(\mathbf{z}) - g(\mathbf{z}))d\mathbf{z} \geq 0$, then $\int_S (f(\mathbf{z}) \log \frac{f(\mathbf{z})}{g(\mathbf{z})} d\mathbf{z} \geq 0$.

If we put $f(\mathbf{z}) = k(\mathbf{z}|\mathbf{y}, \boldsymbol{\Theta})$ and $g(\mathbf{z}) = k(\mathbf{z}|\mathbf{y}, \boldsymbol{\Theta}')$ then clearly $\int_S (f(\mathbf{z}) - g(\mathbf{z}))d|\mathbf{z} \geq 0$, as $\int_S f(\mathbf{z}) = 1$ and by the laws of probability $\int_S g(\mathbf{z})d|\mathbf{z} \leq 1$. Hence

$$\int_S f(\mathbf{z}) \log \frac{f(\mathbf{z})}{g(\mathbf{z})} d\mathbf{z} = \int_S k(\mathbf{z}|\mathbf{y}, \boldsymbol{\theta}) \log \frac{k(\mathbf{z}|\mathbf{y}, \boldsymbol{\theta})}{k(\mathbf{z}|\mathbf{y}, \boldsymbol{\theta}')} d\mathbf{z} \geq 0 \tag{B.5.1}$$

But

$$\begin{aligned} H(\boldsymbol{\Theta}, \boldsymbol{\Theta}) - H(\boldsymbol{\Theta}', \boldsymbol{\Theta}) &= E[\log k(\mathbf{z}|\mathbf{y}, \boldsymbol{\theta}) \mid \widetilde{p}(\mathbf{z})] - E[\log k(\mathbf{z}|\mathbf{y}, \boldsymbol{\theta}') \mid \widetilde{p}(\mathbf{z})] \quad \text{(B.5.2)} \\ &= \int_S k(\mathbf{z}|\mathbf{y}, \boldsymbol{\theta}) \log k(\mathbf{z}|\mathbf{y}, \boldsymbol{\theta}) - \int_S k(\mathbf{z}|\mathbf{y}, \boldsymbol{\theta}) \log k(\mathbf{z}|\mathbf{y}, \boldsymbol{\theta}') \\ &= \int_S k(\mathbf{z}|\mathbf{y}, \boldsymbol{\theta}) \log \frac{k(\mathbf{z}|\mathbf{y}, \boldsymbol{\theta})}{k(\mathbf{z}|\mathbf{y}, \boldsymbol{\theta}')} \geq 0 \quad \text{(B.5.3)} \end{aligned}$$

APPENDIX C

EM for MPPCA

This Appendix contains the formulas that we use in fitting the MPPCA model by the EM algorithm the complete derivations and proofs for these formulas are given in Tipping & Bishop (1999*a*).

- **E-step**

$$R_{ij} = \frac{q_j p_j(\mathbf{y}_i|j)}{f_k(\mathbf{y}_i)}$$

- **M-step**

$$\begin{aligned}
\widetilde{q_j} &= \frac{1}{N}\sum_i R_{ij} \\
\widetilde{\boldsymbol{\mu}} &= \frac{\sum_i R_{ij}(\mathbf{y}_{ij})}{\sum_i R_{ij}} \\
\widetilde{\mathbf{A}_j} &= \mathbf{S}_j\mathbf{A}_j(\sigma_j^2\mathbf{I} + \mathbf{M}_j^{-1}\mathbf{A}_j^T\mathbf{S}_j\mathbf{A}_j)^{-1} \\
\sigma_j^2 &= \frac{1}{d}tr(\mathbf{S}_j - \mathbf{S}_j\mathbf{A}_j\mathbf{M}_j^{-1}\widetilde{\mathbf{A}}_j^T)
\end{aligned}$$

where

$$\begin{aligned}
\mathbf{S}_j &= \frac{1}{\widetilde{q_j}N}\sum_i^N R_{ij}(\mathbf{y}_i - \widetilde{\boldsymbol{\mu}_j})(\mathbf{y}_i - \widetilde{\boldsymbol{\mu}_j})^T \\
\mathbf{M}_j &= \sigma_j^2\mathbf{I} + \mathbf{A}_j^T\mathbf{A}_j
\end{aligned}$$

where the symbol $\sim$ denotes "new" quantities

VITA

Mohamed Elhafiz Mustafa Musa was born in Omdurman Sudan 1963. He graduated in 1989 with honours degree second class upper from the School of Mathematical Sciences, University of Khartoum. He received his M.Sc. degree in 1996 from the same institute. In 1998 he commenced his Ph.D. study at the Image Processing and Computer Vision group in the Department of Computer Engineering, Middle East Technical University under the supervision of Dr. Volkan Atalay. In 2000 he spent 6 month as a guest at Delft University of Technology, Faculty of Applied Physics, Pattern Recognition group under the supervision of Prof. Robert Duin. He had been an instructor in Ahlia higher college in Sudan from 1990 to 1993, and in Sudan University of Science and Technology from 1993 to 1996. Since 2001 he is working with Çankaya University. His main interest areas are pattern recognition, Principal Component Analysis, mixture modelling and Arabic optical character recognition.