Learning Density Models via Structured Latent Variables

Thesis submitted in accordance with the requirements of the University of Liverpool for the degree of Doctor in Philosophy by

Xi Yang

Department of Electrical Engineering and Electronics
School of Electrical Engineering and Electronics and Computer Science
University of Liverpool

Oct. 2018

Abstract

As one principal approach to machine learning and cognitive science, the probabilistic framework has been continuously developed both theoretically and practically. Learning a probabilistic model can be thought of as inferring plausible models to explain observed data. The learning process exploits random variables as building blocks which are held together with probabilistic relationships. The key idea behind latent variable models is to introduce latent variables as powerful attributes (setting/instrument) to reveal data structures and explore underlying features which can sensitively describe the real-world data. The classical research approaches engage shallow architectures, including latent feature models and finite mixtures of latent variable models. Within the classical frameworks, we should make certain assumptions about the form, structure, and distribution of the data. S-ince the shallow form may not describe the data structures sufficiently, new types of latent structures are promptly developed with the probabilistic frameworks. In this line, three main research interests are sparked, including infinite latent feature models, mixtures of the mixture models, and deep models.

This dissertation summarises our work which is advancing the state-of-the-art in both classical and emerging areas. In the first block, a finite latent variable model with the parametric priors is presented for clustering and is further extended into a two-layer mixture model for discrimination. These models embed the dimensionality reduction in their learning tasks by designing a latent structure called common loading. Referred to as the joint learning models, these models attain more appropriate low-dimensional space that better matches the learning task. Meanwhile, the parameters are optimised simultaneously for both the low-dimensional space and model learning.

However, these joint learning models must assume the fixed number of features as well as mixtures, which are normally tuned and searched using a trial and error approach. In general, the simpler inference can be performed by fixing more parameters. However, the fixed parameters will limit the flexibility of models, and false assumptions could even derive incorrect inferences from the data. Thus, a richer model is allowed for reducing the number of assumptions. Therefore an infinite tri-factorisation structure is proposed with non-parametric priors in the second block. This model can automatically determine an optimal number of features and leverage the interrelation between data and features.

In the final block, we introduce how to promote the shallow latent structures model to deep structures to handle the richer structured data. This part includes two tasks: one is a layer-wise-based model, another is a deep autoencoder-based model. In a deep density model, the knowledge of cognitive agents can be modelled using more complex probability distributions. At the same time, inference and parameter computation procedure are s-traightforward by using a greedy layer-wise algorithm. The deep autoencoder-based joint learning model is trained in an end-to-end fashion which does not require pre-training of the autoencoder network. Also, it can be optimised by standard backpropagation without the inference of maximum a posteriori. Deep generative models are much more efficient than their shallow architectures for unsupervised and supervised density learning tasks. Furthermore, they can also be developed and used in various practical applications.

Key Words: Density models, Latent variable models, Deep density models, Gaussian mixture models, Deep autoencoder.

Acknowledgements

My research and education would not have been possible without the help and support of supervisors, colleagues, and family. First and foremost, I am extremely grateful for the advice and support of my primary supervisor Professor Kaizhu Huang who have helped guide me on both academic and personal decisions in the past several years. His academic rigour and innovative thinking inspire me along the way of my research. His guidance has helped me in all the time of this research and in writing of this thesis. Furthermore, I want to thank my co-supervisor Dr. John Y. Goulermas for his patience and support in supervising me in my research, especially on the professional of writing, whose influence can be seen throughout this dissertation.

Besides my advisor, I extend my thanks to all the advisors and colleagues in the Pattern REcognition & Machine Intelligence Laboratory: Dr. Qiufeng Wang, Mr. Haochuan Jiang, Mr. Shufei Zhang, Mr. Guanyu Yang. They have offered me an intensively positive attitude and the technical and equipment support. Moreover, I would also like to thank Xi'an Jiaotong-Liverpool University for providing a Ph.D. scholarship under Grant PGRS-13-03-17.

My sincere thanks also go to my husband Yuyao Yan, also the most important colleague, who supports me not only in life but also in my research. Without his continuous help and support, I cannot insist on my research.

Last but not the least, my parents and my cat have provided me with the love and support I have needed throughout all of this, and without them, none of this would have been possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Machine learning can be thought of as a set of methods for creating plausible models from observed data. A machine can use such models to describe or predict something about the real world. In fact, future data may be not consistent with the model obtained by training the observed data, so the model is required to give uncertain results. Therefore, the probabilistic framework provides uncertainty about the representation, manipulation, and predictions of models [1]. The probabilistic framework mentioned here is synonymous with Bayesian modelling rather than frequency, which adopts probability theory for learning the parameters and structures of a model from data [2, 3]. Besides, for mathematically understanding, Bayesian modelling can be used both in a discriminative and a generative way.

Density modelling introduced in this dissertation mainly refers to the Bayesian decision method which gives the probability density function.[1] This kind of model is a probabilistic framework for estimating distributions which are used to encode data acquired through experience. Concretely, from Bayesian's point of view, it is necessary to assign a prior distribution to the parameters, which is to give the subjective probability distribution characteristics. The observation data are then used to correct the initial probability distribution, which is achieved by multiplying the likelihood function. Finally, the posterior distribution is obtained according to the Bayesian theory for making a decision. This model is particularly useful when the model cannot be confidently learned due to insufficient data. Especially, density estimation is most potent when involving additional latent/hidden variables [4]. Given a different distribution of latent variables, the structured latent variable can correspond to different concepts, such as categories, data structures, semantics, or behavioural state [5, 6]. In addition, latent variables are also introduced to reduce the dimensions of the data. They can also abstract the underlying features in a large number of observations to make the analysis of the data more accessible. Scientif-

[1]Another discriminative method of statistical machine learning is the discriminant function method. The most important special case is the minimum-distance classifier.

ically, latent variables can link observable data in the real world to symbolic data in the modelled world [7]. Therefore density models via latent structures emerge as one of the central theoretical approaches for designing machines, and also as a central role in data mining, pattern recognition, and artificial intelligence [2, 3]. The primary challenge of density models lies at learning high-dimensional data which have complicated data manifolds such as text, images, and sounds. In essence, it embraces the following aspects: i) capturing the crucial attributes in lower-dimensional space, ii) reducing the free parameters, iii) estimating distributions to fit the complicated data manifold, iv) increasing model flexibility, and v) reducing the learning difficulty (costly inference procedures).

This dissertation explores all the above five aspects. When we deal with high-dimensional data, how to capture its low-dimensional representation is the primary task of learning. In classical approaches, dimensionality reduction techniques are always considered as a pre-processing step, which reveals data structure and explores features for subsequent discriminative or generative learning. Essentially, this independent learning may lead to some problems, i.e., the optimal subspace obtained by the dimensionality reduction may not benefit the most the learning task because these two tasks are solved independently without joint optimisation [8, 9]. Therefore, the question becomes how to capture the crucial attributes in the lower-dimensional space, which refers to the attributes related to subsequent tasks. To this end, the first process can be described as the dimensionality reduction for the particular learning task. Recently, coupled with latent variables, joint frameworks are proposed which combine the dimensionality reduction task with the learning task. In the joint frameworks, the parameters of the two tasks are updated simultaneously, which ensures that the learning task can guide the process of finding the low-dimensional representation.

Reducing the parameters can accelerate the model training. Also, it is an important method to alleviate the problem caused by insufficient training data. When the data patterns are high-dimensional but of low cardinality, the ratio of the sample number to the parameter number in the model will be low.[2] With the limited data, we may not learn a good model due to either over-fitting or underestimation of the noise level [10]. Therefore the subspace derived by the independent dimensionality reduction may even significantly deteriorate the discrimination performance. In a model with multiple latent variables, it is a very effective method to set a common latent variable loading for reducing the free parameters. Especially, when the mixture model adopts a common latent variable loading, this variable can be treated as a dimensionality reduction matrix. This method accelerates training while mitigates the negative effect caused by the limited number of per class samples. Chapter 3.2 mainly discusses the performances on small sample size data and

[2]These kinds of data are denoted as Hughs Phenomelia.

the comparison of computational efficiency with the component mixture model. Also, this simple and elegant framework is appropriate for both shallow and deep models.

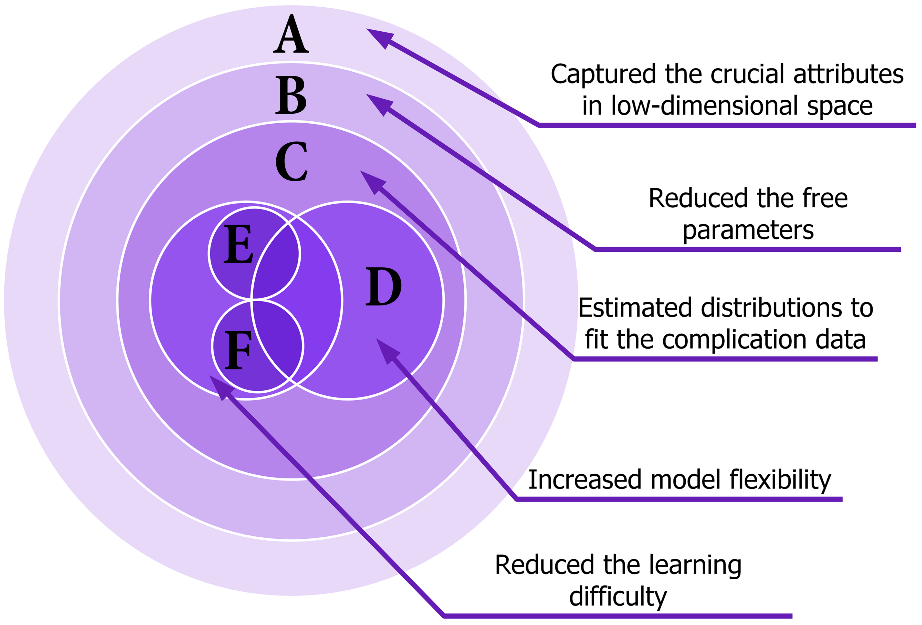
There are two cases of estimating distributions to learn high-dimensional data with complicated data manifolds. In the first case, when objects of multiple types with features of much richer structures are addressed, the single latent factor can be restrictive and often provides poor low-rank matrix approximation [11]. Thus, a new latent factor is needed to absorb the additional scale. Thus a probabilistic model for tri-factorisation is proposed to provide two factors to leverage the interrelation between data and features (in Chapter 4). In the other case, a distribution is usually not enough to represent latent features of a cluster, when the high-dimensional data are just projected by a linear mapping. Therefore, both the multi-layer mixture structured latent variables and the deep-structured latent variables are often used to handle the complicated data manifold. Since the Gaussian mixture model considers that multiple Gaussians can fit any unknown distribution, this thesis proposes a two-layer mixture model to deal with the problem that the data of a single class does not fit well by a multivariate Gaussian (see Chapter 3). Assuming that each class is a generative learning task, we can then use a mixture latent variable model to approximate it.

Moreover, capable of extracting high-level and semantically meaningful features from the sensory data, the deep architecture of density models can create a better prior for complex structured data. In addition, deep neural networks can also be used for feature extraction, and they can further be combined with density model learning. The existing deep models, probabilistic graphical models, and deep neural networks, not only prove that the deep architectures can create a better prior for complex structured data than the shallow density architectures theoretically, but also have practical significance in prediction, reconstruction, clustering, and simulation [12][13].

However, deep density models still encounter computational difficulties in practice due to a large number of free parameters and costly inference procedures [14]. To this end, the common latent variable loading and the greedy layer-wise learning approach are considered to address these concerns (see Chapter 3). Particularly, in the greedy layer-wise learning approach, the same inference procedure is used for each layer. The hot-point, deep neural networks, and the density estimations are further combined. The proposed model (see Chapter 6) manages to exploit the network output as a posterior probability, which simplifies the inference procedures. Meanwhile, the back-propagation can be directly used to reduce the optimisation difficulty.

Finally, there is a trade-off between increasing model flexibility and reducing the learning difficulty. In general, some assumptions need to be made about the form, structure, and/or distribution of data during inferring. However, the more assumptions are made, the simpler it is to perform inference. But if these assumptions are wrong, the inferences drawn from the data would not be correct. To increase flexibility, it is desirable to reduce the number of assumptions and allow for richer models as the understanding of data grows. Therefore, better flexibility can be achieved through the use of non-parametric Bayesian prior on the factor, which liberates the constraint that the dimension of the low-dimensional representation must be fixed [15]. However, the posterior inference is performed in models using the priors of an infinite-dimensional stochastic process since the model is not practical unless posterior distributions can be computed. This increases the difficulty of theoretical analysis of the non-parametric models.

In summary, the relationship among our developed models is described in a stacked Venn diagram Fig. 1.1.


A
B
C
E
D
F
Captured the crucial attributes in low-dimensional space
Reduced the free parameters
Estimated distributions to fit the complication data
Increased model flexibility
Reduced the learning difficulty

A: Latent Variable Models
B: Mixture of Factor Analysers with Common Factor Loading
C: Two-layer Mixture of Factor Analysers with Joint Factor Loading
D: Infinite Non-negative Binary Matrix Tri-factorisation
E: Deep Mixtures of Factor Analysers with Common Loadings
F: Deep Neural Network-based Models

Fig. 1.1: The relationship among the developed models in this thesis.

1.1 Contributions of This Thesis

The research contributions in this dissertation are summarised in the following:

- We propose two frameworks for joint learning of latent variable models: Mixtures of Factor Analysers with Common factor loadings (MCFA) for clustering; Two-layer Mixtures of Factor Analysers with Joint Factor Loading (2L-MJFA) for classification.

  - In comparison with the traditional independent learning (conducted by the latent space and classifier independently), the joint learning methods can learn both the optimal subspace and the parameters for the clustering/classification model simultaneously. The aim is to interact dimensionality reduction with the clustering/classification so that the dimensionality reduction and the clustering/classification can be well matched with each other.
  - For unsupervised learning, we demonstrate that the joint learning with the dimensionality reduction subspace would make the clustering properties clearly reserved. Furthermore, we analyse that the common factor loading could be regarded as the dimensionality reduction matrix, and it also achieves efficient dimensionality reduction that reduces the free parameters (computational time) for high dimensional data.
  - With the above insight, we propose a mixture discriminant model, namely the two-layer joint learning model, which is more suitable for high dimensional but small sample size classification. This relies upon a mixture of mixtures structure, used to capture the complex properties of each class better. Importantly, the common loading matrix mitigates the negative effect caused by the limited number of per class samples.
  - For the two-layer joint learning model, a modified expectation-maximisation algorithm is proposed that consists of two-layer loops, so that the joint learning is conducted very efficiently. The first-layer loop is engaged to estimate the joint parameters that fit the mixture among different classes, whereas the second one trains the mixture components within each class.

- An infinite probabilistic tri-factorisation model is proposed by exploiting the Bayesian nonparametric priors, called Infinite Non-negative Binary Matrix Tri-factorisation (iNBMT).

  - This model is used to address objects of multiple types with features of much richer structures. We show that how to learn the interaction among features and leverage the interrelation between data and features.

  - Specifically, the priors of an infinite-dimensional stochastic process are able to automatically determine an optimal number of features instead of tuning or searching constant parameter by trial and error.
  - Two latent factors are designed in binary form since binary features are cheaper to compute and more compact to store. Binary features can also appear in various types of data. Moreover, no extra constraints are enforced in this model, presenting more appealing features when compared with other models used to generate binary features.
  - We perform posterior inference in models using the priors of an infinite-dimensional stochastic process and compute posterior distributions.

- We develop two kinds of frameworks for deep density models: one is a layer-wise deep architecture called Deep Mixtures of Factor Analysers with Common loadings (DMCFA), the other is a deep neural network-based model.

  - First, we show how to stack multiple layers of joint learning of latent variable models on top of each other by forming a deep belief network. The knowledge of cognitive agents can be modelled using more complex probability distributions, and a deep density model can provide better support for simulating complex data. Thus, the deep model enables a consistent gain in model performance on a wide range of data.
  - By using a greedy layer-wise learning approach, its inference and parameter computation procedure is more straightforward than previous methods. The proposed model is also flexible in estimating the data density by utilising the learnable Gaussian distributions as the priors for each latent unit.
  - We develop a joint learning method via a deep autoencoding framework for supervised learning. The parameters of the deep autoencoder and the Gaussian mixture model are optimised simultaneously in an end-to-end fashion. The joint optimisation can well balance autoencoding reconstruction, latent representation of density estimation, and regularisation terms. This helps the autoencoder to escape the less attractive local optimal solution and further reduce the reconstruction error.
  - There is a difficulty to make significant changes to a well-trained autoencoder by fine-tuning. However, in the end-to-end fashion, pre-training of the autoencoder network is not necessary.

- The introduction of density estimation makes it possible to calculate the output of the network based on the likelihood function instead of the softmax; this enables a probabilistic output which can be more flexibly used in practice.
- The autoencoder-based joint learning model can be optimised by the standard backpropagation. Moreover, the posterior probability is derived from the network and does not require Maximum A Posteriori estimation, so the reasoning of the model is much simpler.

1.2 Summary of Remaining Chapters

Chapter 2 Background In this chapter, we provide a technical overview of the basic building blocks of density models with latent structures. The models include probabilistic Principal Component Analysis, factor analysis, Gaussian mixture model, and the mixtures of factor analysers, Bayesian nonparametric model, deep density model, and deep autoencoding. We also analyse techniques that improve the learning of those mixture models by employing priors of an infinite-dimensional stochastic process. We discuss how to combine shallow mixture models into deeper models via layer-wise stacking. Deep mixture factor analysers present the latest results in deep density models. There is also a brief discussion of another deep generative model, deep autoencoders. At the end of this chapter, we discuss Maximum Likelihood Estimation, Maximum A Posteriori inference, Variational Bayesian inferences, the Expectation-Maximisation algorithm, and hyperparameters.

Chapter 3 Density Model with Finite Mixtures for Unsupervised and Supervised Learning In this chapter, we will detail two models related to joint learning. The first work is the Unsupervised Dimensionality Reduction for Gaussian Mixture Model, demonstrating that joint learning with the dimensionality reduction subspace would make the clustering properties clearly reserved and even clear than independent learning. The second work is to propose a novel joint learning model for classification referred to as a two-layer mixtures of factor analysers with joint factor loading. Explicitly, the model adopts a special two-layer mixture or a mixture of mixtures structure, where each component represents each specific class as a mixtures of factor analysers. Importantly, all the involved factor analysers are intentionally designed so that they share the same loading matrix. Apart from operating as the dimensionality reduction matrix, this largely reduces the parameters and makes the proposed algorithm very suitable to small dataset situations. Additionally, we propose a modified expectation maximisation algorithm to train the proposed model. This chapter is based on the published papers of Yang et al. [16–19].

Chapter 4 Infinite Non-negative Binary Matrix Tri-factorisation for Learning Latent Features In this chapter, we propose a new Bayesian model, termed the infinite Non-negative Binary Matrix Tri-factorisation (iNBMT) model. This can automatically learn both latent binary features and feature numbers, based on the Indian Buffet Process (IBP). It exploits a tri-factorisation process that decomposes the nonnegative matrix into a product of three components: two binary matrices and a non-negative real matrix. In contrast to traditional bi-factorisation, the tri-factorisation can better reveal the latent structures among items (samples) and attributes (features). Specifically, an IBP prior is imposed on two infinite binary matrices while a truncated Gaussian distribution is assumed on the weight matrix. To optimise the model, we develop a modified variational-Bayes algorithm, with iteration complexity one order lower than the recently proposed models [20, 21]. We also demonstrate our approach in two new applications: pre-image restoration and co-clustering. This chapter is based on the published papers of Yang et al. [9, 22].

Chapter 5 A Novel Deep Density Model for Unsupervised Learning In this chapter, we introduce a novel deep density model, referred to as Deep Mixtures of Factor Analysers with Common Loadings (DMCFA), with an efficient greedy layer-wise unsupervised learning algorithm. The model employs a mixtures of factor analysers sharing common component loadings in each layer. The common loadings can be considered to be a feature selection or reduction matrix which makes this new model more physically meaningful. Importantly, sharing common components is capable of reducing both the number of free parameters and computation complexity remarkably. Consequently, DMCFA makes inference and learning rely on a dramatically more succinct model and avoids sacrificing its flexibility in estimating the data density by utilising Gaussian distributions as the priors. This chapter is based on the published papers of Yang et al. [23, 24].

Chapter 6 Deep Neural Network-Based Models via Density Estimation In this chapter, we study a variant of the joint learning model utilising a deep autoencoder to a feature extractor. The goal is to perform hand-writing characters classification through depth generation models and then to generate new samples by categories. The proposed model is referred to as the deep autoencoder-based joint learning model. In this model, the deep autoencoder generates a low-dimensional representation which is further fed into a Gaussian mixture model and controls the reconstruction error for each input data point. Importantly, deep autoencoder-based joint learning is conducted in an end-to-end fashion, which optimises the parameters of the Gaussian mixture model and deep autoencoder simultaneously instead of the decoupled two-stage training, and the inference of the model is kept simple by using the standard backpropagation. Also, the Gaussian mixture model can guide the deep autoencoder projecting latent features onto a Gaussian space and help the autoencoder escape from the less attractive local optimum. Consequently, the joint optimisation, which well balances autoencoding reconstruction and Gaussian mixture model of latent representation, avoids the need for pre-training.

Chapter 7 Conclusion We will then summarise this dissertation and conduct discussions on future work.

We try to make each of these chapters self-contained. Therefore, some critical contents appearing in previous chapters may be briefly reiterated in several chapters, e.g., model definitions or illustrative figures.

Chapter 2

Background

In this chapter, we introduce density models with latent variables and provide the relevant background material. Density models are mainly used to estimate the unobservable probability density function based on the observed data in statistical pattern recognition. The density estimation can offer valuable indications for features of the data such as skewness and multimodality [25]. Among all density models, those with latent variables are particularly interesting. These density models seek to model the latent structure or the latent features of data, with probability density estimation [1, 26]. Moreover, latent variables can be learned to transform the data from its input representation to a more useful latent representation. Algorithms vary in complexity, from simple methods like Probabilistic Principal Component Analysis (PPCA) [27] and Factor Analysis (FA) [28] to complicated methods such as the mixture models and Bayesian nonparametric models. "Deep Density model" derives its name from forming a multi-layer model where the same shallow density model comprises the basic components of each layer. For instance, the well-known Deep Belief Network is formed by the stacking of Restricted Boltzmann Machines [14, 29]. With the development of deep learning, the perception of the density estimation is enhanced by the combination with deep autoencoder, which also makes the establishment of the model simpler, such as the Variational Autoencoders and Deep Autoencoding Gaussian Mixture Model.

In this chapter, we discuss several popular density models with mixtures and Bayesian nonparametric framework, and deep density models with multi-layer and deep autoencoder framework, which will form the basic building blocks of the remaining chapters in this thesis. As the preliminaries, Section 2.1 reviews the density models with latent variables, and also briefly discusses their role in Dimensionality Reduction (DR). Section 2.2 covers many popular finite mixture models for unsupervised learning (clustering) including Gaussian Mixture Model (GMM) and Mixtures of Factor Analysers (MFA) [30, 31]. This section will serve as a building block of the mixture of latent variables models. Section 2.3 reviews a density model with infinite mixtures based on the bi-factorisation framework. The main contribution is the utilisation of the Bayesian nonparametric prior. Section 2.4 and 2.5 review two kinds of deep density models which adopt different deep architectures. The former one is a multi-layer deep model, and the latter one is the combination of the deep autoencoders (AE). Moreover, the latter is a probabilistic model. The multi-layer deep model has the advantage of easier inference and less computational complexity. However, when the number of layers is too big, the performance may not be good enough. With deep AE, this deep model has the advantage of easier optimisation by even adopting the standard back-propagation. Both the model formulation and some novel insights into the improved training will be discussed in each section.

The building block roadmap of this chapter is drawn in Fig. 2.1 which shows clearly the major topics for learning density models with the latent structures. The topics with a tick indicate that they have been discussed or touched in this thesis.

2.1 Density Model with Latent Variables

In statistics, latent variable models are probabilistic approaches parameterized with a set of latent variables and manifest variables [32, 33]. These models are typically grouped depending whether the nature of the manifest and latent variables are discrete or continuous (see Table 2.1) [34, 35]. They are also widely applied to analyse data, especially in the presence of repeated observations, multi-level data, and panel data [4]. The preliminaries

Table 2.1: Different types of latent variable models.

Manifest variables	Latent variables	
	Continuous	Discrete
Continuous	Factor Analysis	Latent Profile Analysis
Discrete	Latent Trait Analysis	Latent Class Analysis

focus on the Factor Analysis and the Latent Profile Analysis, where the manifest variables are continuous and the conditional distribution gave the latent variables, often supposed to be normal [35, 36]. Note that the latent variables are treated as normally distributed for the former variables and a multinomial distribution for the latter.

In general, the most commonly used method for density estimation is the Maximum Likelihood Estimate (MLE). In this way, we can establish a likelihood function $L(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln p(\mathbf{y}_n|\boldsymbol{\mu}, \boldsymbol{\Sigma})$.[1] However, it is computationally difficult to directly cal-

[1]$\mathbf{y}_n$ - N observation data, $\boldsymbol{\mu}$ - mean, $\boldsymbol{\Sigma}$ - covariates

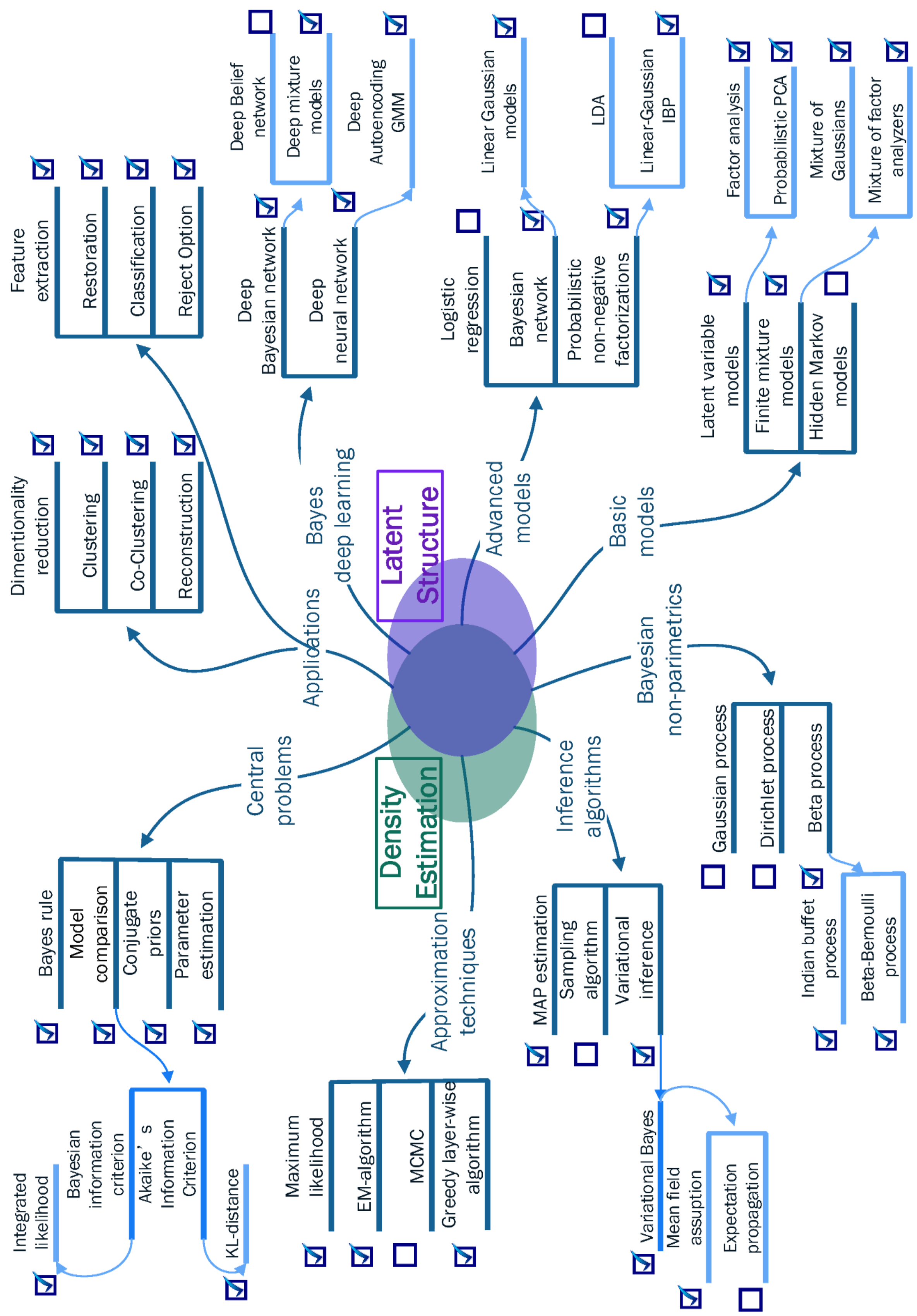


Fig. 2.1: The building blocks of the background.

culate the likelihood functions because of the very high dimensions of Σ. Thus, a set of variables $\mathbf{z}$ is defined to govern multiple $\mathbf{y}$. When the distribution of $p(\mathbf{z})$ is obtained, $p(\mathbf{y})$ can be determined by the joint distribution over $\mathbf{y}$ and $\mathbf{z}$. Typically, the covariates Σ are ruled out. In this setting, $\mathbf{z}$ is assumed to affect the manifest variables (observable variables), but it is not directly observable. Therefore, $\mathbf{z}$ is the so-called the latent variable [36]. Importantly, the introduction of latent variables allows the formation of complicated distributions from simpler components. The latent variable models are widely used in machine learning, data mining, and statistical analysis. As basic models, FA and PPCA are often applied to define an appropriate density distribution of data.

2.1.1 Notation

In density estimation, the density function is assumed to be a simple parametric model, and the parameters of the model are estimated by using a valid training set. We begin by considering the problem of identifying groups/clusters of data points in a multidimensional space. Suppose we have a dataset $\mathbf{y}$ consisting of N observations which have a d-dimensional vector $\{y_1, y_2, \ldots, y_d\}$ of each feature variable. The data involved in models shall follow the identical and independent distribution as assumption, namely, the $i.i.d.$ condition. By introducing latent variables, the manifest distribution $p(\mathbf{y})$ can be signified in terms of a q-dimensional vector $\{z_1, z_2, \ldots, z_q\}$ of latent variables $\mathbf{z}$, where q is a smaller number than d [37]. Through this process, the joint distribution $P(\mathbf{y}, \mathbf{z})$ is decomposed into the conditional distribution of the feature variables given the latent variables and the product of the marginal distribution of the latent variables $p(\mathbf{z})$ [4].

$$P(\mathbf{y}, \mathbf{z}) = p(\mathbf{y}|\mathbf{z})p(\mathbf{z}) = p(\mathbf{z}) \prod_{i=1}^{d} p(y_i|\mathbf{z}) \ . \tag{2.1}$$

As shown in Fig. 2.2, the latent variable models can be graphically represented by a directed graph.

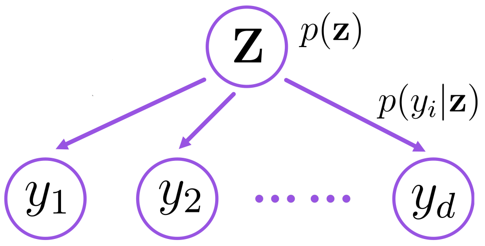


Fig. 2.2: A directed graph of a latent variable models. The factorisation property of Eq. (2.1) can be expressed with a directed graph, in which the feature variables y_i are independent of the latent variables $\mathbf{z}$.

Factor Analysis

Among the four latent variable models in Table 2.1, FA is the most well-known and earliest developed latent variable model [38]. In machine learning, FA is a data dimensionality reduction method exploited for data analysis and processing, and EM algorithms are often engaged to estimate parameters [39]. FA can mine low-dimensional features that still can represent the primary information of many observable attributes from the high-dimensional observations. This low-dimensional representation can generate new data through Gaussian distribution, linear transformation, and error perturbation. As the most common example of a latent variable model, the generative model of FA can be presented as the mapping $f(\mathbf{z}; \mathbf{A})$, a linear function of the latent variable $\mathbf{z}$

$$\mathbf{y} = \mathbf{A}\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon} \ . \tag{2.2}$$

Conventionally, the latent factors $\mathbf{z}$ are defined to be independent to each other and Normal distribution $\mathcal{N}(0, \mathbf{I}_q)$, where $\mathbf{I}_q$ is a $q \times q$ identity matrix. The noise term $\boldsymbol{\epsilon}$ is defined to be a Gaussian with $\boldsymbol{\Psi}$ diagonal variance, and $\boldsymbol{\mu}$ denotes the mean of the data $\mathbf{y}$, which permits the data model to have a non-zero mean. Moreover, the matrix $\mathbf{A} \in \mathbb{R}^{d\times q}$ contains the factor loadings, where $q < d$. Note that this factor loading matrix should be an orthogonal matrix. From this formulation, it can be deduced that the observations are also distributed as the Gaussian $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where the model covariance can be calculated by $\boldsymbol{\Sigma} = \mathbf{A}\mathbf{A}^T + \boldsymbol{\Psi}$.[2] The probability distribution over $\mathbf{y}$-space for a given $\mathbf{z}$ of the form

$$\mathbf{y}|\mathbf{z} \sim \mathcal{N}(\mathbf{A}\mathbf{z} + \boldsymbol{\mu}, \boldsymbol{\Psi}) \ . \tag{2.3}$$

By using the Bayesian rule, the conditional distribution of the latent variables $\mathbf{z}$ given the observations $\mathbf{y}$ is again Gaussian

$$\mathbf{z}|\mathbf{y} \sim \mathcal{N}(\mathbf{V}^{-1}\mathbf{A}^T\boldsymbol{\Psi}^{-1}(\mathbf{y} - \boldsymbol{\mu}), \mathbf{V}^{-1}) \ , \tag{2.4}$$

where $\mathbf{V} = \mathbf{I}_q + \mathbf{A}^T\boldsymbol{\Psi}^{-1}\mathbf{A}$.[3] Note that the size of $\boldsymbol{\Psi}$ is $d \times d$ while $\mathbf{V}$ is a $q \times q$ matrix.

The goal of FA is to best model $\boldsymbol{\Sigma}$ by finding the optimal $\mathbf{A}$ and $\boldsymbol{\Psi}$. Since there are no closed-form analytic solutions, their values must be determined by iterative procedures. The most common method is the Expectation-Maximisation (EM) algorithm [40], which will be detailed in Section 2.2.7. The principal motivation for FA is that the observed variables $\mathbf{y}$ are conditionally independent given the latent variables $\mathbf{z}$ when $\boldsymbol{\Psi}$ is diagonalised. Consequently, FA utilises a small number of latent variables to model the correlations between the elements of the observed variables, and $\boldsymbol{\epsilon}$ can be seen as the variance of each observed variable.

[2] As a result of this parametrisation, $\boldsymbol{\Sigma}$ is invariant under post-multiplication of $\mathbf{A}$ by an orthogonal matrix, which is equivalent to a rotation of the $\mathbf{z}$ coordinate system.

[3] In the equation, $\boldsymbol{\Psi}$ is diagonal variance and $\mathbf{A}$ is orthogonal matrix. Thus there must be an inverse matrix.

Probabilistic Principal Component Analysis

Principal Component Analysis (PCA) is also a classical method of dimensionality reduction. In learning theory, PCA is mainly used to deal with noise or redundant features [41]. PCA is a standardised linear projection in which the most common derivation is to permutate the variance of the projection space by computing the feature vector, but it is not based upon a probability model. In contrast to FA, traditional PCA exploits feature vectors and eigenvalues of covariance to achieve the same intention [42, 43]. Moreover, the maximum likelihood estimate of the parameters in the FA can determine the principal axis of a set of observation vectors. Therefore, PCA is extended into a probability model by introducing a constraint into the covariance matrix of the noise term of the FA model. With the generative model as stated in Eq. (2.2), the PPCA adopts an isotropic Gaussian model $\mathcal{N}(0, \boldsymbol{\sigma}^2 \mathbf{I}_d)$ for the noise term $\boldsymbol{\epsilon}$. The conditional probability distribution is given by

$$\mathbf{y}|\mathbf{z} \sim \mathcal{N}(\mathbf{A}\mathbf{z} + \boldsymbol{\mu}, \boldsymbol{\sigma}^2 \mathbf{I}_d) \; . \tag{2.5}$$

The observation covariance model is specified by $\mathbf{A}\mathbf{A}^T + \boldsymbol{\sigma}^2 \mathbf{I}_d$. Also using Bayes rule, the posterior distribution of the latent variables is obtained by

$$\mathbf{z}|\mathbf{y} \sim \mathcal{N}(\mathbf{V}^{-1}\mathbf{A}^T(\mathbf{y} - \boldsymbol{\mu}), \boldsymbol{\sigma}^2 \mathbf{V}^{-1}) \; , \tag{2.6}$$

where we have $\mathbf{V} = \mathbf{A}^T\mathbf{A} + \boldsymbol{\sigma}^2 \mathbf{I}_d$.

2.1.2 Dimensionality Reduction (DR)

As the latent variable model, the general motivation for FA and PPCA is also to transform the data into some lower-dimensionality representation. In DR, the primary role of the latent variables is to allow a complicated distribution over observed variables constructed from simpler conditional distributions. Importantly, the dimensionality of latent variables is always lower than that of the observable variables. Therefore, the higher-dimensional observable variables are reduced to the low-dimensional latent variables to represent a model.

DR has been one central research topic in information theory, pattern recognition, and machine learning [44, 45]. Apparently, the performance of many learning models significantly relies on DR: successful DR can largely improve various approaches in clustering and classification. When applied to high-dimensional data, existing research approaches often try to reduce the dimensionality first and then input the reduced features to other available models. There are many mature dimensionality reduction methods at present [31, 46–48], such as, linear methods - linear discriminant analysis (LDA), factor analyser (FA), and non-linear method - kernel principal component analysis (kernel-PCA). From the comparison of PCA and PPCA, it is more natural to consider the dimension reduction process according to the distribution of latent variables conditioned on the observation. In PPCA, the principal axes can be found incrementally through observing each iteration.

2.1.3 Discussion

Although PPCA utilises a simple constraint to the standard FA model by the use of the isotropic noise covariance, both of them achieve the goal of DR by finding low-dimensionality representations. In essence, there are significant differences between the two methods:

- The original observed variable is in a different position in two cases: FA exploits a linear combination of the original observed variables as the new factors, and PPCA engages the linear combination of the principal components as the original observed variables.
- Importantly, FA is covariant under component rescaling, while PPCA is covariant under the raw data axis rotating.
- Compared with PCA, FA estimates a low-dimensional loading factor matrix when constructing the model, while PCA converts the observed variables into the same-dimensional representation and then reduces the dimensionality. FA can be considered as an extended version from PCA.

2.2 Finite Mixture Models

In statistics, a mixture model is a density model which utilises the mixing components as the latent variables for representing the presence of subpopulations within an overall population. The finite mixture model is central to latent profile analysis, a typical latent variable model, which is usually engaged in dealing with the case where the data in the same set contain multiple different distributions or the same distributions with different parameters [30, 49]. In contrast to the methods introduced in the previous section, the mixture model needs to consider not only the distribution of components but also the number of components, where both components and mixing quantities are treated as latent variables. Consequently, there are two core problems in the finite mixture model, the density established of the mixed component and the parameter estimation of the mixed model. This finite mixture model is a convex combination of more than one density function and comes up with an expression of heterogeneity in a finite number of latent classes, such as GMM and MFA [30, 50].

GMM refers to a model produced by a linear combination of multiple Gaussian distribution functions. In theory, multiple Gaussian distributions can fit any types of distributions. Therefore, it is meaningful to decompose a set of data into several models based on Gaussian probability density functions [51, 52]. MFA presents improved density estimators that are modelled by allowing each Gaussian in the mixture to be represented in a different lower-dimensional manifold. In particular, MFA simultaneously performs clustering and DR of the data [31, 53].

2.2.1 Notation

By considering a mixture of C multinomial distributions, the density of $\mathbf{y}$ could be modelled as a finite mixture model

$$P(\mathbf{y};\boldsymbol{\theta}) = \sum_{c=1}^{C} \pi_c p(\mathbf{y};\boldsymbol{\theta}_c), \quad where \quad c = 1, \dots, C\ . \tag{2.7}$$

Conventionally, c is the index of mixed components, and $\pi_c = p(c)$ is the probability of mixing components, also called mixing proportions. $p(\mathbf{y};\boldsymbol{\theta}_c)$ is the density function of the component, depending on the parameter vector $\boldsymbol{\theta}_c$.

These distributions are referred to as components of this model (or sub-populations of observations) describing the d-variate density function [54]. The mixture distribution can be expressed by a simple graph, as shown in Fig. 2.3. These components are assumed

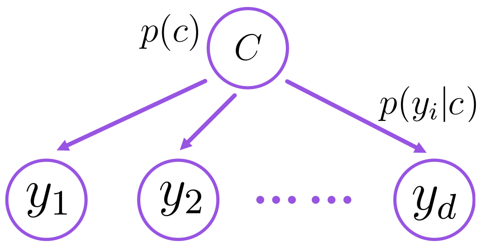


Fig. 2.3: A simple mixture model is expressed in terms of a Bayesian network.

to follow the same parametric family of distributions, but with different parameters, $\boldsymbol{\theta}$. $\boldsymbol{\theta}_c$ denotes the observations' parameters which are associated with the component c. For instance, the mixture components follow Gaussian distributions, and then each component has different means and variances. Additionally, the parameters (means and variances) are random variables in a Bayesian setting, and the prior probability distributions $p(\mathbf{y}|c;\boldsymbol{\theta}_c)$

are placed over the variables [4][36]. $\pi_c = p(c)$ denotes the C mixture weights satisfying the requirement that probabilities sum to 1.

Then, a mixture of latent variables model can be obtained by combining ingredients from the technical of mixture models with the idea of latent variables [55]. Consequently, the joint distribution $P(\mathbf{y}, \mathbf{z})$ is derived as follows and the corresponding Bayesian network is shown in Fig. 2.4.

$$P(\mathbf{y}, \mathbf{z}) = \sum_{c=1}^{C} p(\mathbf{y}|p(c), \mathbf{z})p(c)p(\mathbf{z}|c) = \sum_{c=1}^{C} \pi_c p(\mathbf{z}_c) \prod_{i=1}^{d} p(y_i|\mathbf{z}) \ . \tag{2.8}$$

This mixture model is explicitly computed with discrete latent variables. Hence, the integral is replaced by a sum.

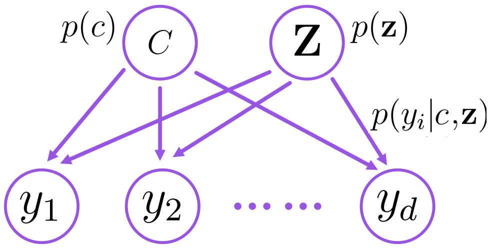


Fig. 2.4: A mixture of latent variables model is graphically expressed by the Bayesian network. The feature variables y_i are conditionally independent of the given mixture weights c and latent variables $\mathbf{z}$.

2.2.2 Gaussian Mixture Model

GMM is a fundamental and widely used model, such as clustering and foreground detection [30, 56]. As a widespread use of clustering algorithm, GMM exploits several Gaussian distributions with deferent parameters as the parametric model. It typically uses the EM algorithm for training. First, the multivariate Gaussian of the d-dimensional observations $\mathbf{y}$ can be defined as follows

$$\mathcal{N}(\mathbf{y}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\big(-\frac{1}{2}(\mathbf{y}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{y}-\boldsymbol{\mu})\big) \ , \tag{2.9}$$

where the mean vector is denoted by $\boldsymbol{\mu}$ and the covariance matrix is $\boldsymbol{\Sigma}$. GMM is a simple extension of the Gaussian model, and GMM engages the weighted sum of several Gaussian models to estimate the distribution of data. According to the central limit theorem, it is reasonable to assume the mixture model is Gaussian [57, 58]. Of course, any distribution can also be defined for the component model. However, more conveniences could be offered in the case of GMM. In addition, it is noted that any probability distribution can be approximated when a sufficient number of mixed components are used. A Gaussian mixture model with C mixed components is defined as the following probability density function

$$P(\mathbf{y};\boldsymbol{\theta}) = \sum_{c=1}^{C} p(c)\mathcal{N}(\mathbf{y};\boldsymbol{\mu}_c,\boldsymbol{\Sigma}_c)\ . \tag{2.10}$$

Here, the parameter set $\boldsymbol{\theta}$ includes $p(c)$, $\boldsymbol{\mu}_c$, and $\boldsymbol{\Sigma}_c$, where $p(c)$ is the weights of the associated mixing proportion and $\sum_{c=1}^{C} p(c) = 1$. The estimation of this probability density is to find $\boldsymbol{\theta}$ of each variable.

2.2.3 Mixtures of Factor Analysers

Traditionally, PPCA can present a low-dimensional representation as a pre-processing step to promote the well-known mixtures of Gaussians [28]. However, as the pre-step, DR ignores the subsequent density estimation tasks, and hence may miss certain information critical for the followed tasks; this approach would easily lead to suboptimal performance. Motivated from the above issue, the idea of joint learning is proposed, which can get the appropriate low-dimensional representation as guided by the DR. Then a joint learning approach is introduced which is referred to as the mixtures of factor analysers. MFA is a globally nonlinear latent variable model which is considered as an extension of the traditional FA. To separate the observations independently into c non-overlapping components, the MFA approach is modelled as

$$\mathbf{y} = \sum_{c=1}^{C} \mathbf{A}_c\mathbf{z} + \boldsymbol{\mu}_c + \boldsymbol{\epsilon}_c \quad \text{with probability } \pi_c\ (c = 1,\ldots,C)\ , \tag{2.11}$$

where $\boldsymbol{\mu}_c$ is a d-dimensional mean vector associated with the component c; $\mathbf{A}_c$ is a $d \times q$ factor loading matrix of the c^{th} component, where $q < d$; and $\pi_c = p(c)$ denotes the weights of the associated mixing proportion. Each (unobservable) factor $\mathbf{z}$ is distributed independently by a normal distribution with zero mean and a $q \times q$ identity covariance matrix, $\mathcal{N}(0,\mathbf{I}_q)$. The noise model $\boldsymbol{\epsilon}_c$ is also independent and assumed as a Gaussian distribution with zero mean and a q-dimensional diagonal covariance matrix, $\mathcal{N}(0,\boldsymbol{\Psi}_c)$. Given the latent variables and the component indicator variables, the conditional distribution of observations $p(\mathbf{y}|c,\mathbf{z})$ follows a Gaussian distribution which has mean $\mathbf{A}_c\mathbf{z} + \boldsymbol{\mu}_c$ and variance $\boldsymbol{\Psi}_c$.

By defining a joint distribution over visible and latent variables $p(\mathbf{y}|c,\mathbf{z})p(\mathbf{z}|c)p(c)$, and the density of the observed data $P(\mathbf{y};\boldsymbol{\theta})$ is obtained by summing up all mixture components. The corresponding distribution of the observed variables is then obtained by marginalising over the latent variables

$$p(\mathbf{y}|c) = \int_{\mathbf{z}} p(\mathbf{y}|c,\mathbf{z})p(\mathbf{z}|c)d\mathbf{z} = \mathcal{N}(\mathbf{y};\boldsymbol{\mu}_c,\mathbf{A}_c\mathbf{A}_c^T + \boldsymbol{\Psi}_c)\ , \tag{2.12}$$

$$P(\mathbf{y};\boldsymbol{\theta}) = \sum_{c=1}^{C} \pi_c p(\mathbf{y}|c,\mathbf{z})p(\mathbf{z}|c) = \sum_{c=1}^{C} \pi_c \mathcal{N}(\mathbf{y};\boldsymbol{\mu}_c,\mathbf{A}_c\mathbf{A}_c^T + \boldsymbol{\Psi}_c)\ , \tag{2.13}$$

where $\boldsymbol{\theta}$ denotes the model parameter vector. It consists of the mixture weight π_c, the mean of the component μ_c, the factor loading $\mathbf{A}_c$ and the covariance of component $\boldsymbol{\Psi}_c$ [53]

$$\boldsymbol{\theta}_c = \begin{pmatrix} \pi_c \\ \\ \mu_c \\ \\ \mathbf{A}_c(vec) \\ \\ \boldsymbol{\Psi}_c(diag) \end{pmatrix}_{c=1}^{C} . \tag{2.14}$$

Therefore, MFA can be considered as a globally nonlinear latent variable model, where C Gaussian factors are fitted on the data.

2.2.4 Inference

For make inference on the density models, we first discuss MLE to obtain the parameters. It can be seen from Section 2.2.5 that the objective function is the logarithm of the sum, which is difficult to expand. Moreover, the optimisation problem is difficult to perform partial bias processing. An iterative method, i.e., the EM algorithm, is typically used.

2.2.5 Maximum Likelihood

After the "best-fitting" distribution (Eq. (2.13)) is obtained, the parameters can then be estimated for that distribution. MLE is a common technique for estimating the parameters of a probability distribution. In other words, MLE can be known as to maximise the sample likelihood by estimating the parameters. These unknown parameters are contained in a vector $\boldsymbol{\theta}$ (as Eq. (2.14)). Loosely speaking, the goal of MLE is to maximise a likelihood function of the sample data. Suppose we have a sample of independent and identically distributed (iid) random variables, $\{\mathbf{y}_1;\mathbf{y}_2;\ldots;\mathbf{y}_N\}$, which is described by the MFA model (Eq. (2.13)). The basic likelihood function is defined as

$$P(\mathbf{y};\boldsymbol{\theta}) = \sum_{c=1}^{C} \pi_c \prod_{n=1}^{N} \mathcal{N}(\mathbf{y}_n;\boldsymbol{\mu}_c,\mathbf{A}_c\mathbf{A}_c^T + \boldsymbol{\Psi}_c)\ . \tag{2.15}$$

Instead of maximising the product, ones usually exploit its logarithm version. It will be e-quivalent to maximise the log-likelihood since the logarithm is an increasing function [53]

$$\mathcal{L}(\boldsymbol{\theta}) = \log P(\mathbf{y};\boldsymbol{\theta}) = \sum_{c=1}^{C}\sum_{n=1}^{N} \log\left\{\pi_c \mathcal{N}(\mathbf{y}_n|\boldsymbol{\mu}_c, \mathbf{A}_c\mathbf{A}_c^T + \boldsymbol{\Psi}_c)\right\} , \tag{2.16}$$

$$\hat{\boldsymbol{\theta}} = \arg\max_{\theta} \mathcal{L}(\boldsymbol{\theta}) . \tag{2.17}$$

Here, $\hat{\boldsymbol{\theta}}$ denotes the estimated parameters when the empirical log-likelihood has a maximum value. In general, we cannot get the exact parameters of the model that have been solved by $\frac{\partial\mathcal{L}}{\partial\boldsymbol{\theta}}$. It is therefore necessary to adopt an iterative method, typically the EM algorithm, that is, to exploit the expectation-maximisation algorithm to maximise the likelihood function.

2.2.6 Maximum A Posteriori

Maximum A Posteriori (MAP) is also a method for estimating variables in the probability distributions setting. MAP is closely related to MLE, which can be regarded as regularisation of MLE. However, MAP is more interested in a posterior distribution, rather than the likelihood. For inference, MAP usually comes up with the Bayesian setting, and the posterior $q(\cdot)$ over the components can be found by recalling Bayes rule

$$q(\mathbf{z}, c|\mathbf{y}) = q(\mathbf{z}|\mathbf{y}, c)q(c|\mathbf{y}) , \tag{2.18}$$

$$q(c|\mathbf{y}) = \frac{p(\mathbf{y}|c)p(c)}{\sum_{h=1}^{C} p(\mathbf{y}|h)p(h)} \propto p(\mathbf{y}|c)p(c) . \tag{2.19}$$

More concretely, the posterior over the latent factors is also a multivariate Gaussian density on $\mathbf{z}$ given $\mathbf{y}$ and c

$$q(\mathbf{z}|\mathbf{y}, c) = \mathcal{N}(\mathbf{z}; \boldsymbol{\kappa}_c, \mathbf{V}_c^{-1}) , \tag{2.20}$$

where

$$\begin{aligned} \mathbf{V}_c^{-1} &= \mathbf{I} + \mathbf{A}_c^T\boldsymbol{\Psi}_c\mathbf{A}_c , \\ \boldsymbol{\kappa}_c &= \mathbf{V}_c^{-1}\mathbf{A}_c^T\boldsymbol{\Psi}_c^{-1}(\mathbf{y} - \boldsymbol{\mu}_c) . \end{aligned} \tag{2.21}$$

A point estimate can be made by selecting the component c by maximising a posterior probability

$$\hat{\boldsymbol{c}} = \arg\max_{c} p(c)p(c|\mathbf{y}) . \tag{2.22}$$

Then, the likelihood in MLE (Eq. (2.17)) is replaced by the posterior. Consequently,

$$\hat{\boldsymbol{\theta}}_{MAP} = \arg\max_{\theta} \sum_{c=1}^{C}\sum_{n=1}^{N} \log\{p(\mathbf{y}|c)p(c)\} = \hat{\boldsymbol{\theta}}_{MLE} . \tag{2.23}$$

2.2.7 Expectation-Maximisation Algorithm

In many applications, the parameters of these density models are determined by the maximum likelihood, which typically adopts the expectation-maximisation (EM) algorithm [59–61]. First, the latent variables must be clarified using the EM algorithm. In density models which have the incomplete data or latent variables, EM algorithms make the parameter estimation possible [60]. In this section, we will show that EM is a powerful and elegant method to find the maximum likelihood solutions for models with latent variables. As the general-purpose iterative strategy, the EM algorithm alternates between two steps. Given the current model, the expected step (step E) is used to predict the probability distribution over completions of missing data. There is usually no need to establish a probability distribution over these completions explicitly, and it is often necessary to calculate "expected" sufficient statistics over completions. The maximisation step (M-step) is used for re-estimating the model parameters by utilising these completions, which can be thought of as "maximisation' of the expected log-likelihood of the data [60].

When the observing variable $\mathbf{y}$, latent variable $\mathbf{z}$, and parameter $\boldsymbol{\theta}$ are given, the mathematical expression can be formulated as follows

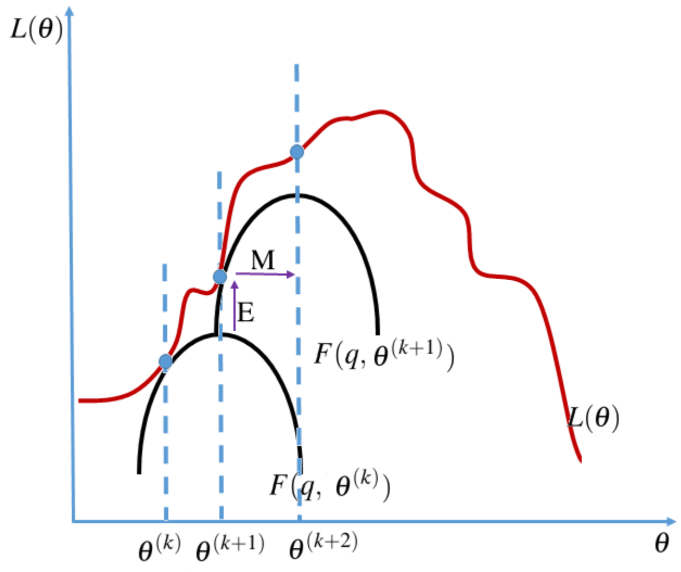
$$\textit{E-step:}\quad Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(k)}) = \mathbb{E}_{q(\mathbf{z}|\mathbf{y};\boldsymbol{\theta}^{(k)})}\big[\log\int_z p(\mathbf{y},\mathbf{z}|\boldsymbol{\theta}^{(k)})d\mathbf{z}\big]\ ; \tag{2.24}$$

$$\textit{M-step:}\quad \boldsymbol{\theta}^{(k+1)} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(k)})\ . \tag{2.25}$$

The convergence of the latent variable-based EM algorithm can be theoretically guaranteed [60, 61]. Based on the Jensen inequality, it is easy to prove that the EM algorithm repeatedly constructs new lower bounds $F(q(\cdot),\boldsymbol{\theta})$ where $q(\cdot)$ denotes the posterior of the latent variable.

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) &= \log\int_z p(\mathbf{y},\mathbf{z}|\boldsymbol{\theta})d\mathbf{z} = \log\int_z q(\mathbf{z}|\mathbf{y};\boldsymbol{\theta})\frac{p(\mathbf{y},\mathbf{z}|\boldsymbol{\theta})}{q(\mathbf{z}|\mathbf{y};\boldsymbol{\theta})}d\mathbf{z}\\ &\geq \int_z q(\mathbf{z}|\mathbf{y};\boldsymbol{\theta})\log\frac{p(\mathbf{y},\mathbf{z}|\boldsymbol{\theta})}{q(\mathbf{z}|\mathbf{y};\boldsymbol{\theta})}d\mathbf{z} = F(q(\cdot),\boldsymbol{\theta})\ , \qquad (2.26)\\ F(q(\cdot),\boldsymbol{\theta}) &= \int_z q(\mathbf{z}|\mathbf{y};\boldsymbol{\theta})\log p(\mathbf{y},\mathbf{z}|\boldsymbol{\theta})d\mathbf{z} - \int_z q(\mathbf{z}|\mathbf{y};\boldsymbol{\theta})\log q(\mathbf{z}|\mathbf{y};\boldsymbol{\theta})d\mathbf{z}\\ &= \mathbb{E}_{q(\mathbf{z}|\mathbf{y};\boldsymbol{\theta})}\big[\log p(\mathbf{y},\mathbf{z}|\boldsymbol{\theta})\big] + H(q(\cdot))\ . \qquad (2.27)\end{aligned}$$

In short, the process of EM is to calculate a lower bound function of the current latent variables distribution according to the current parameters and then to obtain new parameters by optimising this function. The loop will be continued until it converges (as shown in Fig. 2.5).



$L(\boldsymbol{\theta})$
M
E
$F(q, \boldsymbol{\theta}^{(k+1)})$
$L(\boldsymbol{\theta})$
$F(q, \boldsymbol{\theta}^{(k)})$
$\boldsymbol{\theta}^{(k)}$
$\boldsymbol{\theta}^{(k+1)}$
$\boldsymbol{\theta}^{(k+2)}$
$\boldsymbol{\theta}$

Fig. 2.5: Illustration of EM algorithm. The E step is to calculate q by fixing $\boldsymbol{\theta}^{(k+1)}$, and then raise the lower bound from $F(q;\boldsymbol{\theta}^{(k+1)})$ to $F(q;\boldsymbol{\theta}^{(k+2)})$, where $F(q;\boldsymbol{\theta}^{(k+2)})$ is equal to $\mathcal{L}(\boldsymbol{\theta})$ at $\boldsymbol{\theta}^{(k+1)}$. M step is to migrated from $\boldsymbol{\theta}^{(k+1)}$ to $\boldsymbol{\theta}^{(k+2)}$ when q is fixed, and then to find the maximum $F(q;\boldsymbol{\theta}^{(k+2)})$.

2.2.8 Unsupervised Learning and Discussion

One main application of latent models is model-based clustering. Clustering is a method of putting similar data samples together. Probabilistic unsupervised learning builds a generation model that is able to describe clustering. The idea is that all the similar data points belonging to a cluster (all from the same distribution) are more or less equivalent, and any differences between them are accidental [62]. Mixture model-based clustering not only gives a class label for a cluster sample but also generates the probability that each sample belongs to the distribution of a certain model component. The question about the appropriate clustering method and the number of clusters can be converted into questions about how to select the model.

Compared with traditional clustering methods (e.g. k-means), GMM can offer higher flexibility, since the subordination of data points in clustering is not only related to neighbours but also dependent on the shape of clusters. It is worth mentioning that the shape of the multivariate Gaussian distribution is determined by the covariance of each cluster.

Unsupervised dimensionality reduction is often performed together with GMM, while inappropriate DR may deteriorate the GMM's discriminating performance. Although M-FA can be formed by imposing specific constraints on the covariance matrix of each cluster of GMM, the loading matrix of the MFA can be regarded as a DR matrix which cannot be achieved by the standard GMM. Such settings exactly optimise a DR together with the parameters of the learning model. Moreover, joint learning could lead to improvement in comparison with the traditional unsupervised method.

- Joint learning interacts the learning task with DR, so the DR space and the clustering/classification space can be well matched with each other.

- It could lead to much better clustering performance by capturing the crucial attributes in a low-dimensional space.

2.3 Infinite Latent Variable Models

In the finite model, described in the previous two sections, two parameters need be determined. They are the number of mixed components and the dimensions of the low-dimensional representation. To this end, a method is proposed that does not need to specify the cluster number. In the case of adding new samples to the observation set, the number of clusters generated may increase [63, 64]. This method is called the Bayesian nonparametric (BNP) method, which is widely applied to supervised/unsupervised learning (regression, classification, and clustering), latent variable modelling, and image segmentation [65–68]. Although non-parametric methods are referred to as the infinite model, in fact, they can utilise a limited number of sub-arrays in the dimensions of parameters to account for limited observed samples. The choice of the dimension depends on the distribution of samples so that the complexity of the model can be adapted to the data.

For the latent variable modelling, the popular method is to assume a BNP prior on the latent variable - the mixing proportion. As the background, the current state-of-the-art models will be introduced in this section, and the proposed new model able to determine the clusters and latent dimensions will be presented in Chapter 4.

2.3.1 Notation

In the machine learning community, there are many BNP priors based on random infinite-dimensional objects [69]. The current popular BNP priors include:

- The Gaussian process is mainly used for regression and classification. Its structure may change as the sample changes [70].

- The Dirichlet process (Chinese restaurant process) and related priors are used to deal with clustering, which continuously assigns new data into the corresponding clustering [71].

- Beta process (Indian buffet process) and related priors are mainly used for compressed sensing problems and feature extraction, which allow over-lapping properties between objects [72]

In this section, we focus on the Indian buffet process (IBP) of the beta process. IBP provides the prior for the infinite binary matrix so that each row of the matrix has only a finite number of 1. This is the most commonly used model when each object has multiple different attributes, and these attributes can be shared among different objects [72, 73].

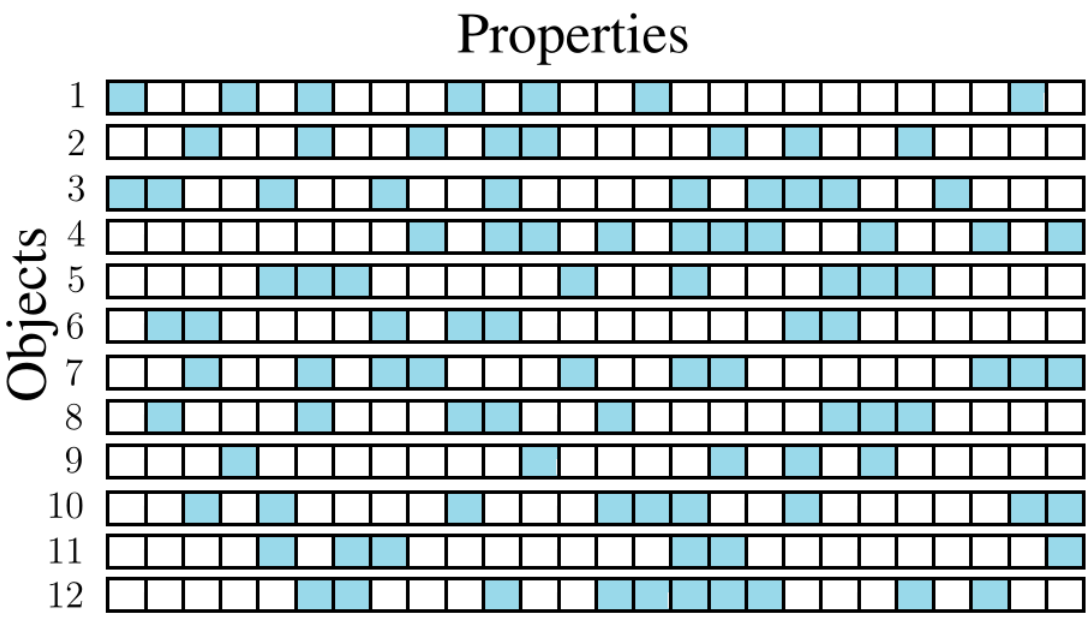


Fig. 2.6: A binary matrix generated by Indian Buffet Process diagram: each object encounters an array consisting of infinite attributes.

A diagram of this process is shown in Fig. 2.6 where the rows of the matrix correspond to objects and the columns correspond to properties. If an object has an attribute, the corresponding column element is 1 (the blue block). IBP can be simply described as N customers orderly entering a buffet restaurant which arranges infinite dishes. A customer can take a $Poisson(\alpha)$ number of dishes. The i^{th} entered customer chooses the dish k with probability $\frac{m_k}{i}$, where m_k is the number of customers selecting this dish previously, and can take $Poisson(\frac{\alpha}{i})$ new dishes after the end of all previously sampled dishes. Note that the order of customers is exchangeable.

2.3.2 Linear-Gaussian IBP Model

The linear-Gaussian IBP model is the simplest example of how to engage IBP as a priori for an unsupervised learning model. As a linear-Gaussian latent feature model, the observed data are still linear functions of the properties, but its hidden layer features are binary. This model is mainly used to generate the grayscale image which consists of a vector of real-valued pixel intensities. The generated model can be written in the form of linearly superimposing different features (visual elements, $\mathbf{Z}$) and additive Gaussian noise $\boldsymbol{\epsilon}$

$$\mathbf{Y} = \mathbf{A}\mathbf{Z} + \boldsymbol{\epsilon} \; . \tag{2.28}$$

Here $\mathbf{Y}$ are the generative images with a D-dimensional vector of pixel intensities, and $\mathbf{A}$ is a $D \times K$ linear projection matrix which can also be considered as a matrix of weights. Essentially, this is a form of binary factor analysis; the feature matrix is reduced to the binary matrix $\mathbf{Z}$ by the DR matrix $\mathbf{A}$ [72]. Thus in a finite model, the i^{th} image $\mathbf{y}_i$ is generated from a Gaussian distribution with mean $\mathbf{A}\mathbf{z}_i$, and covariance matrix $\boldsymbol{\sigma}_y^2\mathbf{I}$, where $\mathbf{z}_i$ is a K-dimensional binary vector, and $\mathbf{I}$ is the identity matrix. In matrix notation, the conditional probability distribution of $\mathbf{Y}$ is Gaussian with mean $\mathbf{A}\mathbf{Z}$ and covariance matrix $\boldsymbol{\sigma}_Y^2\mathbf{I}$. A prior on $\mathbf{A}$ is defined to be a Gaussian with $\boldsymbol{\sigma}_A^2\mathbf{I}$ variance. This prior is conjugate to the likelihood which makes it possible to integrate out the model parameters $\mathbf{A}$. The equation can be rewritten as follows

$$p(\mathbf{Y}|\mathbf{Z}, \boldsymbol{\sigma}_Y, \boldsymbol{\sigma}_A) = \frac{\exp\left\{ -\frac{1}{2\boldsymbol{\sigma}_Y^2} tr\left(\mathbf{Y}^T(\mathbf{I} - \mathbf{Z}(\mathbf{Z}^T\mathbf{Z} + \frac{\boldsymbol{\sigma}_Y^2}{\boldsymbol{\sigma}_A^2\mathbf{I}})^{-1}\mathbf{Z}^T)\mathbf{Y}\right)\right\}}{(2\pi)^{ND/2}\boldsymbol{\sigma}_Y^{(N-K)D}\boldsymbol{\sigma}_A^{KD}|\mathbf{Z}^T\mathbf{Z} + \frac{\boldsymbol{\sigma}_Y^2}{\boldsymbol{\sigma}_A^2\mathbf{I}}|^{D/2}} \; . \tag{2.29}$$

We can extend this to the infinite model with well-defined prior of binary feature $\mathbf{Z}$ which has an unbounded number of columns. Consequently, the infinite number of active features K_+ means K is unbounded, which is learned from data while remaining finite with probability one. By rearranging the non-zero columns of $\mathbf{Z}$, we can specify $K \to \infty$ and write the IBP prior as follows

$$p([\mathbf{Z}]) = \frac{\alpha^{K_+}}{\prod_{h>0} K_h} e^{-\alpha H_N} \prod_{k=1}^{K_+} \frac{(N - m_k)!(m_k - 1)!}{N!} \; , \tag{2.30}$$

where $H_N = \sum_{j=1}^{N} \frac{1}{j}$ denotes the N^{th} harmonic number, and K_h represents the number of non-zero rows. Moreover, K_h and m_k are both irrelevant to the objects sequence, which proves that $p([\mathbf{Z}])$ is an infinitely exchangeable distribution. This model assumes that there are some unknown number of visual elements. Each image is generated by selecting which visual elements the image has.

2.3.3 Variational Bayes Inference

For this model, two basic inference algorithms are commonly applied:

- Sampling algorithms (Gibbs sampling), a family of iterative procedures, derive each random variable by sampling it from the conditional distribution of the given remaining ones.

- Variational inference uses a tractable distribution to approximate an intractable posterior distribution. In general, the parameters of a tractable distribution are chosen by minimising the distance between the approximate distribution and the true posterior.

In this section, we mainly introduce the variational Bayes method. On the one hand, the model includes latent variables; on the other hand, unlike the sampling algorithm, the output of the variational Bayes (VB) inference is a distribution, not a sample. In particular, mean field theory, the only condition of the VB inference, assumes a model $q(\mathbf{Z}) = \prod_i q(\mathbf{z}_i)$ (all of the variables are assumed to be independent) to approximate the posterior probability $p(\mathbf{Z}|\mathbf{Y})$.

If we are setting the KL divergence as the measure of distance, finding $q(\mathbf{Z})$ turns into an optimisation problem $q(\mathbf{Z}^*) = \arg\max_{q(\mathbf{Z})\in Q} KL(q(\mathbf{Z})||p(\mathbf{Z}|\mathbf{Y}))$, where $KL(\cdot)$ is a function about $q(\mathbf{Z})$, and $q(\mathbf{Z}) \in Q$ is a function satisfied the mean field theory. They constitute a functional. The variation extreme is derived from a functional, just as differentiation extreme is derived from a function. The derivation of Evidence Lower Bound Objective (ELBO) [74] is expressed as follows

$$\begin{aligned} KL\big(q(\mathbf{Z})||p(\mathbf{Z}|\mathbf{Y})\big) &= \int q(\mathbf{Z}) \log \frac{q(\mathbf{Z})}{p(\mathbf{Z}|\mathbf{Y})} d\mathbf{Z} = -\int q(\mathbf{Z}) \log \frac{p(\mathbf{Z},\mathbf{Y})}{q(\mathbf{Z})p(\mathbf{Y})} d\mathbf{Z} \\ &= \int q(\mathbf{Z}) \big[\log q(\mathbf{Z}) + \log q(\mathbf{Y})\big] d\mathbf{Z} - \int q(\mathbf{Z}) \log p(\mathbf{Z},\mathbf{Y}) d\mathbf{Z} \quad (2.31) \\ &= \log p(\mathbf{Y}) + \int q(\mathbf{Z}) \log q(\mathbf{Z}) d\mathbf{Z} - \int q(\mathbf{Z}) \log p(\mathbf{Z},\mathbf{Y}) d\mathbf{Z} \ . \end{aligned}$$

Because of $KL(q(\mathbf{Z})||p(\mathbf{Z}|\mathbf{Y})) \geq 0$, $F(q(\mathbf{Z})) = \int q(\mathbf{Z}) \log q(\mathbf{Z}) d\mathbf{Z} - \int q(\mathbf{Z}) \log p(\mathbf{Z},\mathbf{Y}) d\mathbf{Z}$ can be considered as the lower bound of $\log p(\mathbf{Y})$. Then we can obtain the following expression

$$KL\big(q(\mathbf{Z})||p(\mathbf{Z}|\mathbf{Y})\big) = \log p(\mathbf{Y}) + F(q(\mathbf{Z})) \ . \quad (2.32)$$

Since the logarithmic evidence $\log p(\mathbf{Y})$ is a likelihood function that does not depend on Z, it can be regarded as a constant. Therefore, in order to minimise the KL divergence, our goal can be converted to maximise the lower bound. According to the mean field theory, the form of the lower bound can be expanded to

$$\begin{aligned} F(q(\mathbf{Z})) &= \int \log q^*(\mathbf{z}_j) d\mathbf{z}_j - \sum_{i:i\neq j} \log q(\mathbf{z}_i) - constant \\ &= \int q(\mathbf{z}_j) \frac{\log q^*(\mathbf{z}_j)}{q(\mathbf{z}_j)} d\mathbf{z}_j - \sum_{i:i\neq j} \log q(\mathbf{z}_i) - constant \\ &= -KL\big(q(\mathbf{z}_j)||q^*(\mathbf{z}_j)\big) + \prod_{i:i\neq j} H\big(q(\mathbf{z}_i)\big) - constant\ , \end{aligned} \tag{2.33}$$

where $H(.) \geq 0$ is the information entropy, and $q^*(\mathbf{z}_j) = \frac{\exp(\mathbb{E}^{i\neq j}[p(\mathbf{Z},\mathbf{Y})])}{normaliseconstant}$. Consequently, maximising $F(q(\mathbf{Z}))$ only needs to make $-KL(q(\mathbf{z}_j)||q^*(\mathbf{z}_j)) = 0$, that is, to make $q(\mathbf{z}_j) = q^*(\mathbf{z}_j)$.

In addition, VB's lower bound equation $F(q(\mathbf{Z}))$ can be further deduced in an expectation form $F(q(\mathbf{Z})) = \mathbb{E}_{q(\mathbf{Z}|\mathbf{Y})}[\log p(\mathbf{Y},\mathbf{Z})] + H(q(\mathbf{Z}))$, which is the same form as EM's ELBO equation (see Eq. (2.27)). Thus, the EM-like iteration method can also be employed to update the lower bound, which is a popular application of variational Bayes to latent variable models called Variational Bayes EM (VBEM). VBEM is considered to be a special case of the generalised EM by assuming that mean-field requires component independence. In the approximating distribution, the latent variables and parameters are independent, and often there are additional variational approximations within either the latent variables or the parameters.

2.4 Deep Density Model

In the above sections, we defined the shallow architectures and also exhibit inferences to optimise a single layer. We now consider how to extend the shallow models by defining the deep density models. First, the deep architectures need to have many layers of latent variables. Second, parameters should be learned by the efficient greedy layer-wise algorithms. This section introduces the density models with latent variables that explicitly show how inferences of the models correspond to operations in a single layer of a deep architecture. Therefore, the deep model is a directed generative model and exploits the layer-wise training procedure as approximations. In this section, one deep density model will be described, Deep Mixtures of Factor Analysers (DMFA) who adopts an MFA in each hidden layer [31, 75]. The observation vector and the first hidden layer are treated as an MFA model (see Chapter 2.2.3) and learning parameters are manipulated by this unsupervised method. After fixing the first-layer parameters, the priors of next layer MFAs are replaced by sampling the hidden units of the current-layer MFA. The same scheme can be extended to train the following layers. Compared with the shallow models with the same scale mixtures (Collapse Models), the deep models have fewer free parameters and a simpler inference procedure. On the one hand, the components of adjacent layers share the parameters; on the other hand, too many mixtures may make the overall objective function of a shallow model too complicated.

2.4.1 Deep Density Model via Greedy Layer-wise Learning

The latent variable models are usually expressed as a shallow architecture which has one single visible layer and one single hidden layer, such as a directed acyclic graph (see Fig. 2.7). As shown in Chapter 3.2.1, shallow architectures have achieved good perfor-

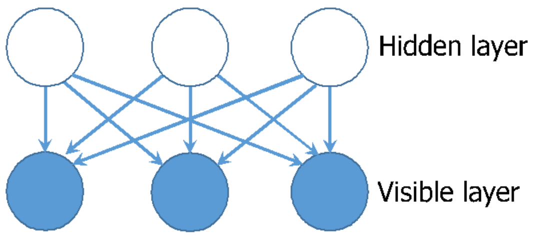


Fig. 2.7: A directed acyclic graph represents a shallow density model which has only one hidden layer.

mances in practice [18]. Nonetheless, shallow architectures may have a serious problem, i.e., it is very inefficient for them to deal with the complex structural data or large amounts of hidden units [76]. In contrast, the deep architectures have multiple hidden layers to help in learning better representations of the complicated hidden units. A directed deep architecture is illustrated in Fig. 2.8. Also, deep architectures reduce the need for hand-crafted features which is a very time-consuming process requiring expert knowledge. However, it is difficult to train all layers at once when a generative model has multi-layered architectures [77].

To this end, the greedy layer-wise procedures are developed for training deep generative models by using the same criterion for optimising each layer starting from the bottom and for transferring the problem upwards to the next layer (which is called the layer-wise learning algorithm [77]). In this sense, in addition, to providing a framework for building more complex probability distributions [78], the latent variable models could be an ideal criterion for a deep architecture. Besides, the deep architectures can also be used for a finite mixture model, which means each sub-population is assumed to come from the manifest variables having more than one distributions.

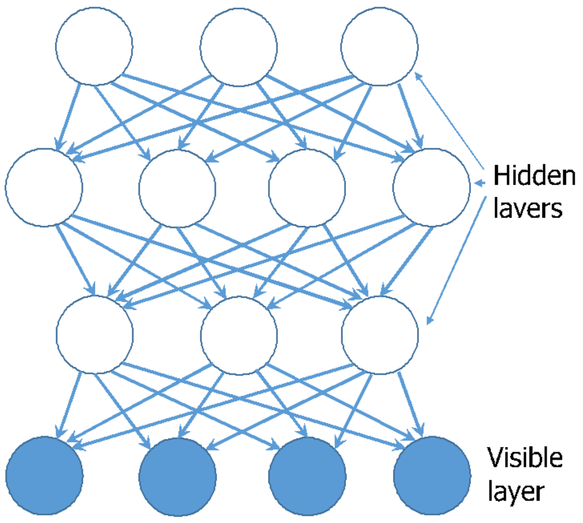


Fig. 2.8: A multi-layer directed acyclic graph represents a deep density model which has three hidden layers and one visible layer.

2.4.2 Deep Generative Models

Tang et al. proposed a deep density model utilising a greedy layered unsupervised learning algorithm, referred to as deep mixtures of factor analysers (DMFA) [75].[4] Unlike previous methods, this model is a directed graphical model which has been developed by adopting MFA. In particular, DMFA extends the same scheme as MFA to train each hidden layer and takes the EM algorithm to maximise the log-likelihood in learning [50, 61]. Its inference and parameter computation procedure is more straightforward than previous methods. However, it is a highly parameterised model where the number of parameters may not be manageable. In real applications, overfitting could become a severe problem in DMFA, since it adopts multiple different factor loadings. Additionally, the latent factors are specified to follow a multivariate standard normal prior, which may limit its flexibility and hinder the accurate estimation of the density.

Having formulated the MFA model, we now show how to construct MFA into a deep architecture. In a shallow model, each FA in MFA has an isotropic Gaussian prior in its factor, as well as a Gaussian posterior over each training sample. However, the posterior is generally non-Gaussian when a posterior is aggregated for many training samples. If we replace the prior of each FA with a separate mixed model, this mixture model can learn to model an aggregated posterior rather than an isotropic Gaussian. The sketches are shown in Fig. 2.9. Therefore, it can improve a variational lower bound on the log

[4]The greedy layer-wise algorithm is a generative model with many layers of hidden variables.

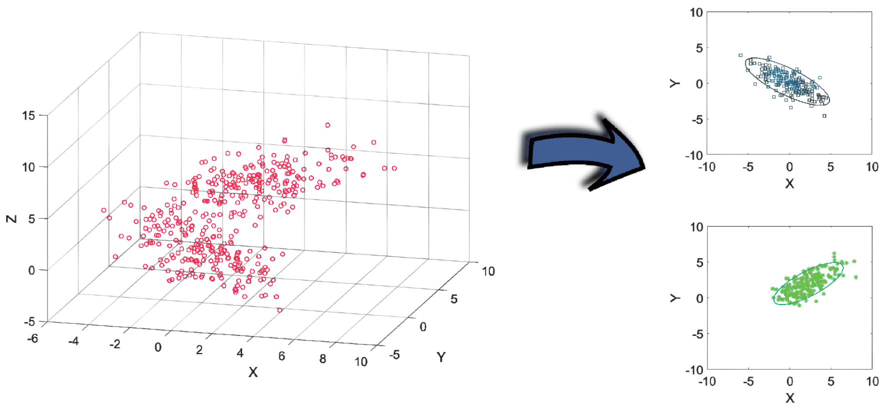


Fig. 2.9: The sketch of a mixture of the DMFA's higher layer for clustering and dimensionality reduction. **Left**: The aggregated posterior of a component in the lower layer is not a Gaussian distribution. **Right**: The higher layer has an ability to model a better aggregated posterior of the lower layer.

probability of the training data [31][75]. According to this method, Tang et al. [75] construct a DMFA model by replacing the FA in the mixture with an MFA model and even substitute the FA of the next mixtures. The graphical model of two-layer DMCFA is visualised in Fig. 2.10. Importantly, the pivotal method is used to sample the data regarding the posterior distributions of the current layer, which can be treated as the training data for the next layer.

The observations $\mathbf{y}$ and the factors $\mathbf{z}$ in the first hidden layer are treated as the MFA, and the first-layer parameters $\boldsymbol{\theta}^{(1)}$ are adopted by this unsupervised method. The aggregated posterior over $\mathbf{z}$ factor with a specific component c is given by $\frac{1}{N}\sum_{n=1}^{N} p(\mathbf{z}^n, c_n = c|\mathbf{y}_n)$. For the second layer, $\mathbf{z}^{(1)}$ and $\hat{c}$ are treated as training data by sampling posterior distribution (Eq.(2.20), Eq.(2.22)). Then a more powerful MFA prior replaces the standard multivariate normal prior

$$p(\mathbf{z}|c) = P_{MFA}(\mathbf{z}_c^{(1)}; \boldsymbol{\theta}_{cs}^{(2)}) \ . \tag{2.34}$$

The same scheme can be extended to train the following layers.

In the second layer, some new symbols need to be defined: $\mathbf{z}^{(1)}$ is a q-dimension vector as the data input to the second layer; $\boldsymbol{\theta}_{c_s}^{(2)}$ emphasises a new parameter vector in the second layer, which is specific to the component c of the first-layer MFA.[5] The layer factors are denoted as a d-dimension vector $\mathbf{z}^{(2)}$; s is a new sub-component indicator variable, and the total number of sub-components is S satisfying $S = \sum_{c=1}^{C} M_c$; $m_c = 1, \ldots, M_c$

[5]The superscript represents which layer these variables belongs to

denotes the number of sub-components corresponding to the c^{th} first-layer component; The mixing proportions in the second layer $p(s) = \pi_s^{(2)}$ are defined as $p(c_s)p(s|c_s)$, where $\sum_{s=1}^{S} \pi_s^{(2)} = 1$ and c_s denote the sub-components corresponding to the c component. Then, the DMFA prior is written as follows

$$p(\mathbf{z}; c) = p(c)p(m_c|c)p(\mathbf{z}|m_c) \ . \tag{2.35}$$

The density of vectors $\mathbf{z}^{(1)}$ follows the joint density over $\mathbf{z}^{(2)}$ and s

$$p(\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, s) = p(\mathbf{z}^{(1)}, c|\mathbf{z}^{(2)}, s)p(\mathbf{z}^{(2)}|s)p(s) \ , \tag{2.36}$$

$$p(\mathbf{z}^{(1)}, c|s, \mathbf{z}^{(2)}) = \mathcal{N}(\mathbf{z}^{(1)}; \mathbf{W}_s^{(2)}\mathbf{z}^{(2)} + \boldsymbol{\mu}_s^{(2)}, \boldsymbol{\Psi}_s^{(2)}), \quad p(\mathbf{z}^{(2)}|s) = \mathcal{N}(0, \mathbf{I}) \ . \tag{2.37}$$

Here, the new parameter vector $\boldsymbol{\theta}_{c_s}^{(2)}$ consists of $\mathbf{W}_s^{(2)} \in \mathbb{R}^{q \times d}$, $\boldsymbol{\Psi}_s^{(2)} \in \mathbb{R}^{q \times q}$, $\boldsymbol{\mu}_s^{(2)} \in \mathbb{R}^q$, $\mathbf{z}^{(2)} \in \mathbb{R}^d$.[6] Specifically, since every s is just allowed to belong to one and only one c, we obtain the Gaussian density on the observed data $\mathbf{y}$ given $\mathbf{z}^{(1)}$ and c

$$p(\mathbf{y}|c, \mathbf{z}^{(1)}) = \mathcal{N}(\mathbf{y}; \mathbf{W}_c^{(1)}\mathbf{z}^{(1)} + \boldsymbol{\mu}_c^{(1)}, \boldsymbol{\Psi}_c^{(1)}) \ , \tag{2.38}$$

where, $\mathbf{W}_c^{(1)} \in \mathbb{R}^{p \times q}$, $\boldsymbol{\Psi}_c^{(1)} \in \mathbb{R}^{p \times p}$, $\boldsymbol{\mu}_c^{(1)} \in \mathbb{R}^p$, $\mathbf{z}^{(1)} \in \mathbb{R}^q$ denote the first layer parameters.

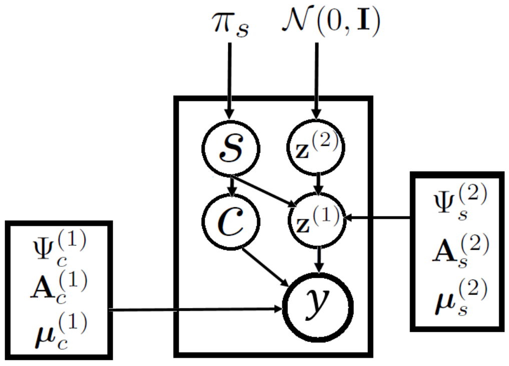


Fig. 2.10: Graphical models of a two-layer DMFA model. DMFA is a deep directed graphical model utilising the multi-layer factor analysers which are developed by adopting an MFA model in each hidden layer.

2.4.3 Inference

For inference, the posterior distribution is computed in a similar fashion with Eq. (2.18) and Eq. (2.20)

$$\begin{aligned} q(\mathbf{z}^{(2)}, s|\mathbf{z}^{(1)}, c) &= q(\mathbf{z}^{(2)}|\mathbf{z}^{(1)}, c, s)q(s|\mathbf{z}^{(1)}, c) \\ &= \mathcal{N}(\mathbf{z}^{(2)}; \boldsymbol{\kappa}_{c_s}^{(2)}, \mathbf{V}_{c_s}^{(2)^{-1}}) \ , \end{aligned} \tag{2.39}$$

[6] d denotes the d-dimension subspace of second layer, where $d < q$.

where

$$\mathbf{V}_{c_s}^{(2)^{-1}} = \mathbf{I}_d + \mathbf{W}_{c_s}^{(2)^T} \mathbf{\Psi}_{c_s}^{(2)^{-1}} \mathbf{W}_{c_s}^{(2)} \ ,$$
$$\boldsymbol{\kappa}_{c_s}^{(2)} = \mathbf{V}_{c_s}^{(2)^{-1}} \mathbf{W}_{c_s}^{(2)^T} \mathbf{\Psi}_{c_s}^{(2)^{-1}} (\mathbf{z}^{(1)} - \mathbf{W}_{c_s}^{(2)} \boldsymbol{\mu}_c^{(2)}) \ .$$

Here, the subscript emphasises the sub-component s which is specific to component c of the first layer, and $\mathbf{I}_d$ is a d-dimensional identity matrix. The posterior over the components can be found as follows

$$q(s|\mathbf{z}^{(1)}, c) \propto p(\mathbf{z}^{(1)}, c|s)p(s) \ , \tag{2.40}$$
$$\hat{\boldsymbol{s}} = \arg\max_s p(s)q(s|\mathbf{z}^{(1)}) \ . \tag{2.41}$$

For the second layer, $p(\mathbf{z}^{(1)})$ and $\hat{c}$ are treated as input data and initial labels when the first layer parameters are fixed and Eq. (2.17) is maximised. According to Equation 2.34, the DMFA formulation seeks to find a better prior $p(\mathbf{z}|c) = P_{MFA}(\mathbf{z}^{(1)}|\hat{c}; \boldsymbol{\theta}_{c_s}^{(2)})$.

Given the new data vectors $\{\mathbf{z}_1^{(1)}; \mathbf{z}_2^{(1)}; \ldots; \mathbf{z}_q^{(1)}\}$, maximising Eq. (2.15) with respect to the second layer parameters is equivalent to maximising the density function $P(\mathbf{z}^{(1)}|\hat{c}; \boldsymbol{\theta}_{c_s}^{(2)})$. The basic likelihood objective function of the second layer is defined as

$$P(\mathbf{z}^{(1)}|\hat{c}; \boldsymbol{\theta}_{c_s}^{(2)}) = \sum_{s \in c} \pi_s \prod_{i=1}^{q} \{\mathcal{N}(\mathbf{z}_i^{(2)}|\boldsymbol{\mu}_s^{(2)}, \mathbf{W}_s^{(2)}\mathbf{W}_s^{(2)^T} + \mathbf{\Psi}_s^{(2)})\} \ . \tag{2.42}$$

It is worth noting that, at the second layer, each mixture model of C components derived from the first layer can be updated separately, since S second-layer parameters are non-overlapping and just allowed to belong to one and only one of the first layer components.

$$\hat{\boldsymbol{\theta}}_{c_s}^{(2)} = \arg\max_{\theta_{c_s}^{(2)}} \log P(\mathbf{z}^{(1)}; \boldsymbol{\theta}_{c_s}^{(2)}) \ . \tag{2.43}$$

Despite the good performance in practice, the DMFA model still has many drawbacks. Specifically, this model utilises different loading matrices for different components, which may lead to over-fitting in practical applications. Meanwhile, it also inherits the shortcoming of MFA, that is, assuming that the prior of each potential factor follows a standard Gaussian distribution, which may limit the flexibility and accuracy.

In the deep model, the greedy layer-wise unsupervised algorithm is usually used as an efficient and simple optimisation algorithm to perform inference and learning. In the layer-wise algorithm, the EM algorithm can typically be utilised to estimate the parameters of each mixture in each layer to find a local maximum of the log-likelihood. Importantly, a simple objective function can be fed to the EM algorithm in a layer-wise style instead of the massive objective function of a shallow model with same scale mixtures.

For instance, the expectation log-likelihood of the mixture in the first layer is shown as follows

$$\begin{aligned} Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(k)}) &= \sum_{c=1}^{C} \int_z q(\mathbf{z}, c|\mathbf{y}; \boldsymbol{\theta}_c) \ln p(\mathbf{y}, \mathbf{z}, c|\boldsymbol{\theta}_c) d\mathbf{z} \\ &= \mathbb{E}_{q(\mathbf{z},c|\mathbf{y};\boldsymbol{\theta}_c)} \big[\ln p(\mathbf{y}|c, \mathbf{z}) + \ln p(\mathbf{z}|c) + \ln \pi_c \big] \quad = \mathbb{E}_{q(\mathbf{z},c|\mathbf{y},\boldsymbol{\theta})} [\log \mathcal{L}(\boldsymbol{\theta}_c)] \ . \end{aligned} \tag{2.44}$$

During M-step, the parameters $\boldsymbol{\theta}_c^{k+1}$ (at $(k+1)^{th}$ iteration) are updated by solving the partial differentiation of the expectation log-likelihood equation over each parameter

$$\frac{\partial \mathbb{E}_{q(\mathbf{z},c|\mathbf{y},\boldsymbol{\theta}_{old})}[\log \mathcal{L}(\boldsymbol{\theta}_c)]}{\partial \boldsymbol{\theta}_c} = 0 \ . \tag{2.45}$$

The higher layer has an ability to model a better aggregated posterior of the first layer, with variational inference. Moreover, any increase in the bound will improve the true log-likelihood of the model when the bound is tight. In this sense, it is better to train the deep model than a shallow model.

2.4.4 Collapse Model

It is noted that DMFA can also be collapsed into a shallow form by integrating out the latent factors. According to Eq. (2.12), we obtain the collapse model after the first-layer factors $\mathbf{z}^{(1)}$ are integrated out

$$\begin{aligned} p(\mathbf{y}|\mathbf{z}^{(2)}, s) &= \int_{\mathbf{z}^{(1)}} p(\mathbf{y}|c, \mathbf{z}^{(1)}) p(\mathbf{z}^{(1)}|s, \mathbf{z}^{(2)}) p(\mathbf{z}^{(2)}|s) d\mathbf{z}^{(1)} \\ &= \mathcal{N}\big(\mathbf{y}; \mathbf{W}_c^{(1)}(\mathbf{W}_s^{(2)}\mathbf{z}^{(2)} + \boldsymbol{\mu}_s^{(2)}) + \boldsymbol{\mu}_c^{(1)}, \mathbf{W}_c^{(1)}\boldsymbol{\Psi}_s^{(2)}\mathbf{W}_c^{(1)^T} + \boldsymbol{\Psi}_c^{(1)}\big) \ . \end{aligned} \tag{2.46}$$

Then the final shallow form is obtained by further integrating out $\mathbf{z}^{(2)}$

$$p(\mathbf{y}|s) = \textstyle\int_{\mathbf{z}^{(2)}} p(\mathbf{y}|\mathbf{z}^{(2)}, s) d\mathbf{z}^{(2)} = \mathcal{N}(\mathbf{y}; \mathbf{m}_s, \boldsymbol{\Sigma}_s) \ , \tag{2.47}$$

$$\mathbf{m}_s = \mathbf{W}_c^{(1)}\boldsymbol{\mu}_s^{(2)} + \boldsymbol{\mu}_c^{(1)}, \quad \boldsymbol{\Sigma}_s = \mathbf{W}_c^{(1)}(\boldsymbol{\psi}_s^{(2)} + \mathbf{W}_s^{(2)}\mathbf{W}_s^{(2)^T})\mathbf{W}_c^{(1)^T} + \boldsymbol{\Psi}_c^{(1)} \ . \tag{2.48}$$

Finally, the marginal density of the shallowed model on observed data $\mathbf{y}$ is then given by a mixture of Gaussians

$$p(\mathbf{y}) = \sum_{s=1}^{S} p(s)p(\mathbf{y}|s) = \sum_{s=1}^{S} \pi_s \mathcal{N}(\mathbf{y}; \mathbf{m}_s, \boldsymbol{\Sigma}_s) \ . \tag{2.49}$$

Conventionally, $\boldsymbol{\theta}_s = \{\pi_s, \boldsymbol{\mu}_s, \mathbf{W}_s, \boldsymbol{\Psi}_s, \mathbf{W}_c, \boldsymbol{\mu}_c, \boldsymbol{\Psi}_c\}_{s=1,c=1}^{S,C}$ represent the parameters of the shallow form of DMFA.

In this case, the posterior probability of the shallowed MFA collapsing from a two-layer DMFA for the s^{th} mixture can be given by $p(s|\mathbf{y}) = \pi_s p(\mathbf{y}|\mathrm{s})/p(\mathbf{y})$. We are also interested in the posterior distribution of the latent factor $\mathbf{z}_s$ which is collapsed to a shallow form

$$\begin{aligned} q(\mathbf{z}^{(2)}, s|\mathbf{y}) &= \mathcal{N}(\mathbf{z}_s^{(2)}; \boldsymbol{\kappa}_s, \mathbf{V}_s^{-1}) \ , \\ \mathbf{V}_s^{-1} &= (\mathbf{W}_s^{(2)}\mathbf{W}_s^{(2)^T} + \mathbf{\Psi}_s^{(2)})^{-1} + \mathbf{W}_c^{(1)^T}\mathbf{\Psi}_c^{(1)^{-1}}\mathbf{A}_c^{(1)} \ , \\ \boldsymbol{\kappa}_s &= \mathbf{W}_s^{(2)^T}\boldsymbol{\mu}_s^{(2)} + \mathbf{V}_s^{-1}\mathbf{A}^{(1)^T}\mathbf{\Psi}^{(1)^{-1}}(\mathbf{y} - \boldsymbol{\mu}_c) \ . \end{aligned} \tag{2.50}$$

2.5 Deep Autoencoding Density Model

This section will introduce another type of deep density models using the framework of the deep AE. We name them as Deep Autoencoder-based Density Model, to mainly distinguish the layer-wise deep density model. Deep Learning is a learning algorithm which is already an important branch of artificial intelligence. Deep learning can be trained in an end-to-end style. Thus it subverts the idea of algorithm design in many fields, such as speech recognition, image classification, text understanding and so on [79–81]. In recent years, due to the improvement of hardware (Graphics Processing Unit (GPU)) and the optimisation method (Back Propagation, BP), deep learning has quickly become the most popular method in the learning community with its concise framework and powerful perception ability. Moreover, in just a few years, it has also been rapidly applied to a wide range of practical applications.

One of the simplest methods of deep learning is to take a hierarchical system of neural networks (NN). If an NN is given, we assume that its output is the same as the input, and then train the adjustment. Its parameters give the weights in each layer. Naturally, we get several different representations of input data (each layer represents a representation), and these representations are features. An AE is a neural network that reproduces the input signal as much as possible. In order to achieve this recurrence, AE must capture the most important factor that can represent the input data, just like PCA, find the main components that can represent the original data (as shown in Fig. 2.11).

However, this structure can only be applied in data compression. On the other hand, building a density model on the latent variable ($\mathbf{z}$) allows the model to achieve more learning goals, such as generating new data, classification, and clustering [13, 82, 83]. To this end, the Variational Autoencoder (VAE) is developed to extend AE to be a generative model. In this section, VAE will be introduced as the preliminary [84], and a state-of-the-art deep autoencoding Gaussian mixture model will also be described [85].

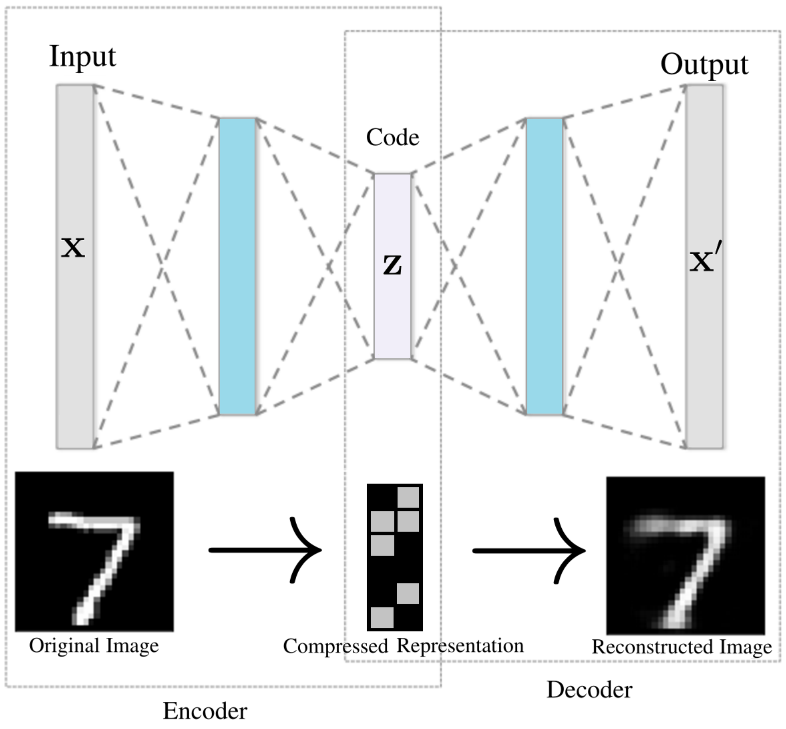


Fig. 2.11: Schematic structure of an autoencoder with 3 fully-connected hidden layers.

2.5.1 Notation

The goal of VAE is to construct a model that generates target data $\mathbf{X}$ from a latent variable $\mathbf{Z}$. More precisely, By assuming that $\mathbf{Z}$ obeys a common distribution (normal distribution or uniform distribution, a model $\mathbf{X} = g(\mathbf{Z})$ can then be learned. This model maps the probability distribution of the target data to the probability distribution of the latent variables. Fig. 2.12 shows the structure of a standard VAE with fully-connected hidden layers. First, the unlabeled data $\mathbf{X}$ are taken as input, and then the features are learned using unsupervised learning. The features are generated by the encoder (Code: $h = \tanh(\mathbf{W}_0\mathbf{X} + b_0)$). After that, two encoders will be connected, one for calculating the latent variable mean $\boldsymbol{\mu} = \mathbf{W}_1\mathbf{X} + b_1$, and the other for calculating the hidden variable variance $\log \boldsymbol{\sigma}^2 = \mathbf{W}_2\mathbf{X} + b_2$. The goal is to make the generated data similar to the metadata. The log-likelihood of observed data $\mathbf{x}_i$ can be written as

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}_i) = KL\big(q_\phi(\mathbf{z}|\mathbf{x}_i)\big) + \mathcal{L}(\boldsymbol{\theta}, \phi; \mathbf{x}_i) \ . \tag{2.51}$$

Here $\mathcal{L}$ represents the evidence of the lower bound, which is usually maximised so as to obtain the best parameters. We can optimise the likelihood indirectly by optimising $\mathcal{L}$.

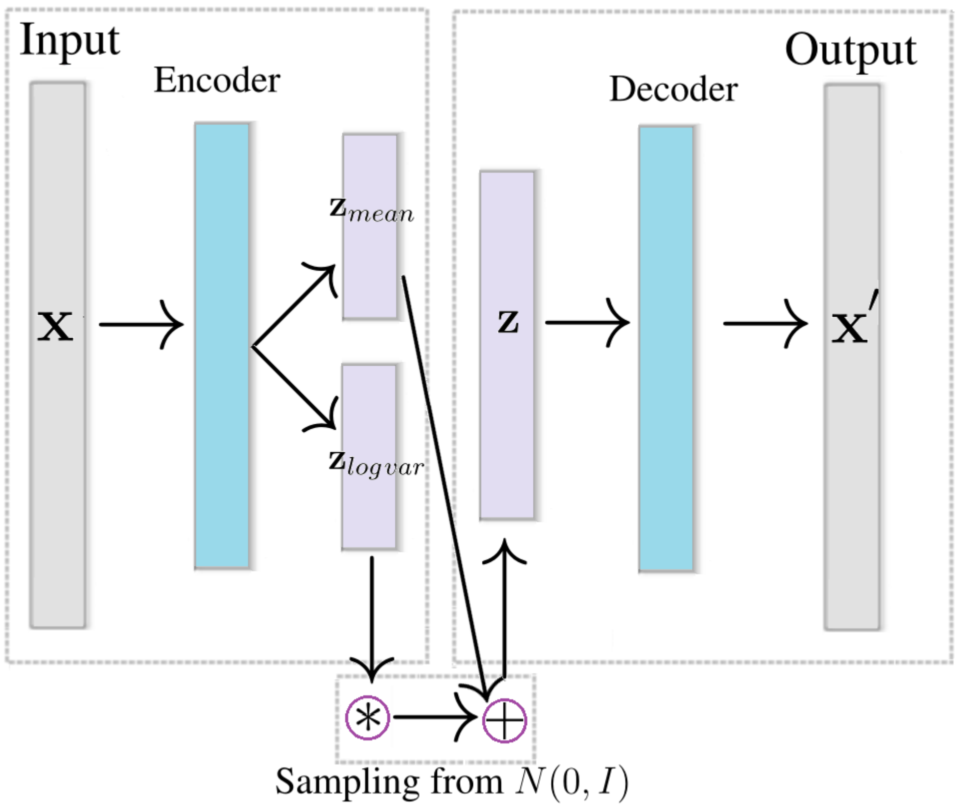


Fig. 2.12: Schematic structure of a variational autoencoder with fully-connected hidden layers.

According to the multiplication formula of probability, $\mathcal{L}$ can be transformed into

$$\mathcal{L}(\boldsymbol{\theta}, \phi; \mathbf{x}_i) = -KL\big(q_\phi(\mathbf{z}|\mathbf{x}_i)\big) + \mathbb{E}_{q_{\boldsymbol{\theta}}(\mathbf{x}_i)}\big[\log p_{\boldsymbol{\theta}}(\mathbf{x}_i)\big] \ . \tag{2.52}$$

It can be observed that the optimised goal can be broken down into two items. Let's first examine the first item, which is a KL divergence. $q_\phi(\mathbf{z}|\mathbf{x}_i)$ is the distribution we would like to learn, which takes each dimension's independent Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$. In fact, in the VAE model, we assume that the posterior distribution of latent variables $p(\mathbf{z}|\mathbf{x}_i)$ is a standard normal distribution $\mathcal{N}(0, \mathbf{I})$. By reasonably selecting the distribution form, this item can be analytically solved. Then the KL divergence between the Gaussian distribution and standard normal distribution can be calculated as

$$-KL(q_\phi(\mathbf{z}|\mathbf{x}_i)) = -0.5\big(1 + \log \boldsymbol{\sigma}_i^2 - \boldsymbol{\mu}_i^2 - \boldsymbol{\sigma}_i^2\big) \ . \tag{2.53}$$

Then, we can look at the second term on the right-hand side: $\mathbb{E}_(q_{\boldsymbol{\theta}}(\mathbf{x}_i))[\log p_{\boldsymbol{\theta}}(\mathbf{x}_i)]$ is the Log-likelihood of posterior probability about $\mathbf{x}_i$.

$$\mathbb{E}_{q_{\boldsymbol{\theta}}(\mathbf{x}_i)}[\log p_{\boldsymbol{\theta}}(\mathbf{x}_i)] \approx \frac{1}{L}\sum_{j=1}^{L} \log p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{z}_j) \ . \tag{2.54}$$

Here, $\mathbf{z}_j$ is not directly sampled from the Gaussian distribution modelled by the decoder but exploits the reparameterisation method. If we only take one sample point at a time, then $\mathbb{E}_(q_{\boldsymbol{\theta}}(\mathbf{x}_i))[\log p_{\boldsymbol{\theta}}(\mathbf{x}_i)] \approx \log p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{z})$. Among them, $\mathbf{z}$ is the sampling point. Fortunately, this is the loss function commonly utilised in neural networks. Therefore, VAE converts the similarity between the generated distribution and the true distribution into a summation of the similarity between the distribution of the latent variable and the normal distribution (the posterior distribution of the hidden variable), and the similarity between the sample of the generated distribution and the true distribution.

2.5.2 Deep Autoencoding Gaussian Mixture Model

Deep Autoencoding Gaussian Mixture Model (DAGMM) is one of the state-of-the-art deep AE-based density models, which utilises a deep AE to generate a low-dimensional representation. It further feeds the compact representation into a GMM for performing density estimation [85]. DAGMM usually consists of two major components, which is illustrated in Fig. 2.13.

- A compression network performs two parts of low-dimensional representations, one is the latent features $\mathbf{z}_c$ for input samples $\mathbf{x}$ by a deep AE, and the other is the reconstruction error features $\mathbf{z}_r$ by measuring the distance between input and output samples $\mathbf{x}'$.

- An estimation network outputs a K-dimensional vector for the soft mixture-component membership $\hat{\gamma}$, which predicts the likelihood/energy in the framework on the low-dimensional representations $\mathbf{z}$.

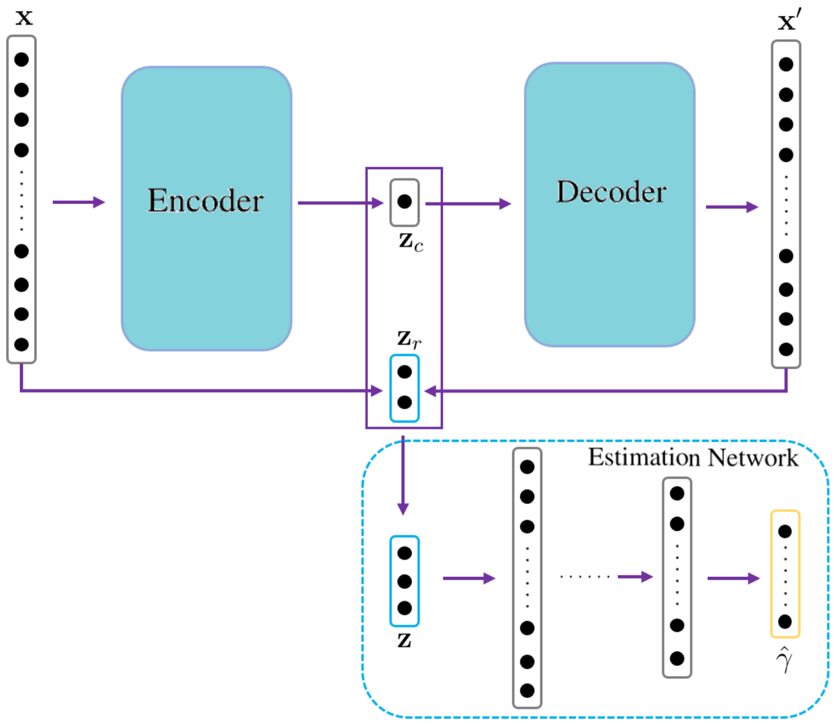


Fig. 2.13: Deep Autoencoding Gaussian Mixture Model: Schematic structure for the compression network combined with the estimation network.

The compression network computes the low-dimensional representation as follows

$$\mathbf{z}_c = h(\mathbf{x}; \boldsymbol{\theta}_e)\ , \qquad \mathbf{x}' = g(\mathbf{z}_c; \boldsymbol{\theta}_d\ ), \tag{2.55}$$

$$\mathbf{z}_r = f(\mathbf{x}, \mathbf{x}')\ , \qquad \mathbf{z} = [\mathbf{z}_c, \mathbf{z}_r]\ , \tag{2.56}$$

where $h(\cdot)$ denotes the encoding function, $g(\cdot)$ denotes the decoding function, and $f(\cdot)$ denotes the function of reconstruction error. All the parameters of the deep AE and multi-layer network are denoted by $\boldsymbol{\theta}$ with different subscripts. $\mathcal{L}(\mathbf{x}, \mathbf{x}')$ is the loss function that characterises the reconstruction error caused by the deep AE.

Given the number of mixture components K, the membership prediction can be obtained from the estimation network outputs as follows

$$p = MLP(\mathbf{z};\boldsymbol{\theta}_p)\ , \qquad \hat{\gamma} = softmax(p)\ , \tag{2.57}$$

$$\hat{\boldsymbol{\phi}}_k = \sum_{i=1}^{N} \frac{\hat{\gamma}_{ik}}{N}\ , \qquad \hat{\boldsymbol{\mu}}_k = \frac{\sum_{i=1}^{N} \hat{\gamma}_{ik}\mathbf{z}_i}{\sum_{i=1}^{N} \hat{\gamma}_{ik}}\ , \tag{2.58}$$

$$\hat{\boldsymbol{\Sigma}}_k = \frac{\sum_{i=1}^{N} \hat{\gamma}_{ik}(\mathbf{z}_i-\hat{\boldsymbol{\mu}}_k)(\mathbf{z}_i-\hat{\boldsymbol{\mu}}_k)^T}{\sum_{i=1}^{N} \hat{\gamma}_{ik}}\ , \tag{2.59}$$

where p is the output of the multi-layered perceptions $MLP(\cdot)$, $softmax(\cdot)$ denotes the soft-max function, and $\hat{\boldsymbol{\phi}}_k$, $\hat{\boldsymbol{\mu}}_k$, $\hat{\boldsymbol{\Sigma}}_k$ are mixing proportion, mean, and covariance for the k^{th} component in GMM.

The loss function of the estimation networks is inferred by the negative log-likelihood with these estimated parameters

$$E(\mathbf{z}) = -\log\Big\{\sum_{k=1}^{K} \hat{\boldsymbol{\phi}}_k \mathcal{N}(\mathbf{z};\hat{\boldsymbol{\mu}}_k,\hat{\boldsymbol{\Sigma}}_k)\Big\}\ . \tag{2.60}$$

In addition, the regularisation $P(\hat{\boldsymbol{\Sigma}}_k) = \sum_{k=1}^{K}\sum_{j=1}^{d}(\hat{\boldsymbol{\Sigma}}_{kjj})^{-1}$ alleviates the singularity problem by penalising small values on the diagonal entries, where d is the low-dimensional representations' dimensions. Given the above, this objective function can be constructed as follows

$$J(\boldsymbol{\theta}_c,\boldsymbol{\theta}_d,\boldsymbol{\theta}_p) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(\mathbf{x},\mathbf{x}') + \frac{\lambda_1}{N}\sum_{i=1}^{N}E(\mathbf{z}_i) + \frac{\lambda_2}{N}P(\hat{\boldsymbol{\Sigma}}_k)\ , \tag{2.61}$$

where λ_1 and λ_2 are the meta parameters which are usually set to $\lambda_1 = 0.1$ and $\lambda_2 = 0.005$.

Since the model is mainly used for unsupervised anomaly detection[7], the reconstruction error features are added as the input to the GMM [86]. In essence, the model takes advantage of the fact that anomalies deviate from the cluster in low-dimensional space and are difficult to reconstruct. Furthermore, anomaly samples can be predicted when their $\mathbb{E}(\mathbf{z})$ is higher than a pre-chosen threshold. Compared with other anomaly detection models, such as Deep clustering network (DCN)and DSEBM-r [87, 88], DAGMM is an end-to-end model which optimises the parameters of deep AE and GMM simultaneously.

2.5.3 Rejection

In real-world applications, some samples may not belong to any known class. Therefore, it might be necessary to refuse to make decisions on these samples, which can further reduce

[7]Anomaly detection is a kind of application of rejection recognition.

the error rate. These rejected samples can be discarded or hold on for more information. This process is called rejection recognition.

DAGMM is mainly applied for the one-class detection by using the value of log-likelihood as the detection criteria. In this case, most of the data are considered normal and modelled in an unsupervised way, and then the abnormal data are detected if the value of log-likelihood is smaller than a pre-defined threshold. Another case is the so-called out-of-distribution detection which is applied for the multi-class detection. It rejects the test samples from different distributions of training data by training a prediction confidence. Recent work has demonstrated that the common multi-class classifiers (neural networks) tend to make highly confident predictions of all test samples, even if they are completely unrecognisable or irrelevant inputs [89–92]. In recent years, the emerging approaches have been proposed to improve the classifier so that such uncertainty can be considered. One seemingly straightforward approach is to enlarge the training set, but the number of out-of-distribution examples can be infinitely many. It keeps an challenge [86, 93] to detect out-of-distribution examples without further re-training networks.

Chapter 3

Density Model with Finite Mixture for Unsupervised and Supervised Learning

A density model with finite mixture usually utilises latent variables to represent the p-resence of sub-populations, e.g., various components, within an overall population, e.g., the mixture model. Typically, the finite mixture models provide a convenient and formal setting for the model-based unsupervised learning, i.e., the Gaussian Mixture Model [30] and the Mixtures of Factor Analysers [50]. These methods can also be used in the model-based supervised learning. For example, when the sub-populations cannot be approximated by a simple or known distribution, a finite mixture model can offer a better fit for each sub-population.

In this chapter, two finite density models will be introduced for unsupervised and supervised learning respectively. More specifically, we will first discuss how to establish a joint learning method which performs the dimensionality reduction and the following learning task simultaneously. We then verify the effectiveness of this model for unsupervised learning, i.e., clustering. Next, we discuss how to reduce the free parameters typically for high-dimensional complicated data. To this end, we propose a latent variable model that uses a hierarchical structure, while assuming a common dimensionality reduction matrix for each component. This model is verified in the setting of supervised learning on various data.

The rest of this chapter is organised as follows: in Section 3.1, a joint learning model is introduced by embedding a common loading matrix in a finite mixture model, highlighting the point that the learned low-dimensionality representations can be calibrated for subsequent learning tasks. Experiments are reported for the joint learning models on several real-world datasets in Section 3.2. In Section 3.3, we develop a mixture discrimination model for the high-dimensional but small sample sized data. Finally, we conclude this chapter in Section 3.4 and also discuss the limitations and future work.

3.1 Unsupervised Dimensionality Reduction for Gaussian Mixture Model

Dimensionality Reduction (DR) has been an important yet active research area in information theory, pattern recognition, and machine learning. Among them are Principal Component Analysis (PCA), Independent Component Analysis (ICA), Fisher Discriminant Analysis (FDA), Latent Dirichlet Analysis (LDA), Maxi-Min Discriminant Analysis (MMDA) [94], and 1-norm based feature selection approach. This is especially the case for high-dimensional data since such data usually contain much redundant information. DR can be engaged to map these high-dimensional data into a low dimensional space, where meaningful or semantic features could be available. Such latent features, better reflecting the relationship within data, can be input to any learning models, e.g., Gaussian mixture model (GMM) [95], and may lead to performance improvement. In the past, there has been a great deal of works in this field [94, 96, 97]. In the context of classification or regression [98], DR could be conducted in the supervised style by utilising certain supervised information (e.g., class labels) so as to find a subspace where different classes of data could be separated as far as possible. These methods include the above mentioned FDA and MMDA. On the other hand, when the class information is not available, DR is performed in an unsupervised way. This family of approaches includes the famous PCA and independent component analysis [44].

In practice, some dimensionality reduction are usually performed independently before the low-dimensional features are fed to available learning models. For example, when GMM is utilised for high-dimensional data, PCA could be conducted beforehand. Then the reduced features are input to a GMM so as to obtain the best parameters. The purpose is both to reduce the computational time for high dimensional data and to find a suitable subspace where better clustering or classification performance could be achieved due to the removal of possible noisy features. In this setting, the optimal subspace and the following optimal parameters of GMM are searched independently. Consequently, the optimal subspace obtained by the independent DR may not be appropriate for the following GMM. This is particularly the case in the context of unsupervised learning, e.g., clustering. In supervised learning, class labels could be used for deriving a good subspace, whilst in unsupervised learning, the principles used for DR (e.g., maximisation of the variance in PCA) may not be appropriate for GMM. Figure 3.1(a) in Section 3.1.3 illustrates the best 2-dimensional subspace obtained by PCA in one synthetic data. Obviously, the original clustering information among data was less obvious after PCA. The detailed discussion can be later seen in the experimental section.

In comparison with the traditional independent learning of DR and GMM (i.e., conducted independently and separately), we propose to learn both the optimal subspace and the parameters for GMM jointly. Specifically, we engage the Mixtures of Factor Analysers (MFA) [99] where a common factor loading is assumed to exist for all latent factors. Importantly, when this special MFA called MCFA is optimised via the modified EM algorithm, the common factor loading could be regarded as the dimensionality reduction matrix, while the mixtures of latent factors can be regarded as GMM. When GMM is used for unsupervised clustering, its joint learning with the DR subspace will make the clustering properties clearly reserved and even clear. To see the advantages, we also show in Figure 3.1(b) of Section 3.1.3 the subspace obtained by the joint learning method. Obviously, it could lead to much better clustering performance, especially compared with PCA. We will also discuss this comparison later in the experimental section. Despite its good properties, the EM algorithm is widely known as a local optimizer, guaranteeing on the global optimum. Hence, the engaged algorithm used in this paper also leads to local-minimum solution. Nonetheless, the experimental results showed that the EM can still generate satisfactory results.

It should be noted that although MFA has been earlier discussed for literature such as [100], it was presented from the viewpoint of data analysis rather than dimensionality reduction. More importantly, the idea of using common loadings, or the joint learning, could also be applied in other mixture models [54]. This presents one important contribution of this section. The rest of this section is organised as follows. First of all, we present the preliminaries used in this section and also briefly review the finite mixture model. In Section 3.1.2, we then introduce a novel MFA model with the common factor loading. The model definition and the optimisation method will be described in turn. In Section 3.1.3, we compare the proposed new joint learning model on five datasets against the other two competitive methods. This work can also be seen in [16, 18] for a short version.

3.1.1 Preliminaries

Probabilistic mixture methods, founded on statistical theory, have become well established in the learning algorithms and have been extensively used in many applications [101, 102]. In the following, we will first introduce Gaussian latent variable models [50], and then we will review the fundamentals of its special case, that is probabilistic PCA [27]. In this part, the notation of this section will be presented with the focus on introducing Gaussian latent variable models. In this model, the marginal and conditional distribution are all subject to Gaussian distribution. The basic mixture model is defined as

$$P(\mathbf{y};\boldsymbol{\theta}) = \sum_{i=1}^{g} \pi_i \prod_{j=1}^{n} P(\mathbf{y}_j \mid \mathbf{z}_{ij}) \ . \tag{3.1}$$

where $P(\mathbf{y})$ is a mixture of g multi-variate normal component distribution of the random variable $\mathbf{Y} = (\mathbf{y}_1, \dots, \mathbf{y}_p)^T$. $\mathbf{z}_{ij}$ is q-dimensional latent variable and $\mathcal{N}(\mathbf{y}_j \mid \mathbf{z}_{ij})$ is the marginal distribution which can be known as a component of this model. Indeed, it also can be seen that the observed p-dimensional data are equivalent to mapping to the q-dimension latent subspace by linear transformation.

The unknown parameter vector $\boldsymbol{\theta}$ consists of the mixture weight π_i, the means of component $\boldsymbol{\mu}_i$, and the covariance of component matrices $\boldsymbol{\Sigma}_i (i = 1, \dots, g)$. This vector can be estimated by maximising the log-likelihood function: $\log \mathcal{L}(\boldsymbol{\theta}) = \sum_{j=1}^{n} \log P(\mathbf{y}_j; \boldsymbol{\theta})$, where $\{\mathbf{y}_j\}$ $(j = 1, \dots, n)$ is an observed random sample set. By using the Expectation-Maximisation (EM) algorithm [100], the local maximisers of log-likelihood function can be obtained by an appropriate root of following equation

$$\partial \mathbb{E}[\log \mathcal{L}(\boldsymbol{\theta})] / \partial \boldsymbol{\theta} = 0 \ . \tag{3.2}$$

With the Bayes theorem, the posterior distribution $P(\omega_i \mid \mathbf{y_j}; \boldsymbol{\theta})$ can be expressed as

$$P(\omega_i \mid \mathbf{y}_j; \boldsymbol{\theta}) = \frac{\pi_i P(\mathbf{y}_j \mid \mathbf{z}_{ij})}{\sum_{h=1}^{g} \pi_h P(\mathbf{y}_j \mid \mathbf{z}_{hj})} \ , \quad i = 1, \dots, g \ ; j = 1, \dots, n \ . \tag{3.3}$$

Here, ω_i represents the i-th latent component category that each sample $\mathbf{y}_j$ belongs to. Note that the category of each sample $\mathbf{y}_j$ is unknown, and the latent variable is the indicator variable ω_i, $\omega_i = \{0, 1\}, \pi_i = P(\omega_i = 1)$. A data point could be assigned to the component that has the highest estimated posterior probability.

The normal mixture model has $m = \frac{p(p+1)}{2}$ parameters for each component covariance matrix. In particular, MFA with the factor-analytic representation of the component covariance matrices as $\boldsymbol{\Sigma}_i = \mathbf{A}_i \mathbf{A}_i^T + \boldsymbol{\Psi}_i$ where $\boldsymbol{\Psi}_i$ is a diagonal matrix and the $p \times q$ matrix $\mathbf{A}$ contains the factor loading [99]. Since $\mathbf{A}$ is an orthogonal matrix, $\mathbf{A}_i$ has $d = \frac{q(q-1)}{2}$ constraints. Hence the number of free parameters in each covariance matrix is $pq + p - \frac{q(q-1)}{2}$. The total number of parameters is

$$m_1 = g - 1 + 2gp + gpq - \frac{gq(q-1)}{2} \ .$$

To reduce further the number of parameters so as to favour efficient optimisation, another popular mixture method-mixture of probabilistic PCA (mPPCA) [43] assumes an isotropic covariance instead of a diagonal matrix covariance of MFA. Each component covariance matrix is defined as $\mathbf{\Sigma}_i = \mathbf{A}_i\mathbf{A}_i^T + \boldsymbol{\sigma}^2\mathbf{I}_p$. The total number of parameters is reduced to

$$m_2 = g + gp + gpq - \frac{gq(q-1)}{2} \ .$$

Even with the isotropic matrix, the number of parameters still may not be manageable when p or q is large. In this section, the novel algorithm provides a great reduction by reducing the parameters in loading matrices. The total number of parameters is further reduce to

$$m_3 = g - 1 + p + gq + pq + \frac{gq(q+1)}{2} - q^2 \ ,$$

with the restrictions

$$\begin{aligned} \boldsymbol{\mu}_i &= \mathbf{A}\boldsymbol{\xi}_i \ ; \\ \mathbf{\Sigma}_i &= \mathbf{A}\mathbf{\Omega}_i\mathbf{A}^T + \mathbf{\Psi} \ ; \\ \mathbf{\Psi}_i &= \mathbf{\Psi} \ , \end{aligned} \tag{3.4}$$

where $\mathbf{A}$ is a $p \times q$ matrix of loadings on latent factors, $\boldsymbol{\xi}_i$ is a q-dimensional vector, and $\mathbf{\Omega}_i$ is a symmetric matrix. In addition to the manageable computational efficiency, when the above special MFA is optimised via the modified EM algorithm, the common factor loading could be regarded as the dimensionality reduction matrix, while the mixture of latent factors can be regarded as a GMM (see Chapter 2.2.2). This exactly achieves the joint learning, leading to much better clustering performance than the independent approaches.

As a linear model, FA decomposes a factor loading to cross a linear subspace within the covariate vector space, making factors have a lower dimension than the covariates. In the following, we will first introduce MFA [50], and then we will review the fundamentals of its special case, that is MCFA [100]. Let $\mathbf{Y} \in \mathbb{R}^{n \times p}$ denote n p-dimensional vectors of feature variables generated by a linear combination of latent variables $\mathbf{Z}$.

3.1.2 Unsupervised Dimensionality Reduction with MCFA

In this section, we will introduce the model Mixtures of Factor Analysers with Common factor loadings (MCFA) so as to learn jointly the dimensionality reduction and the parameters of GMM. We will introduce the model definition first, and then present the involved optimisation algorithm.

Model Description

The latent variable model seeks to generate a p-dimensional observed data vector $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_p)^T$ by linear combination with q-dimensional vector of latent variables $\mathbf{Z} =$

$(\mathbf{z}_{i1}, \ldots, \mathbf{z}_{in})^T$

$$\mathbf{Y} = f(\mathbf{Z}; \mathbf{A}) + \boldsymbol{\epsilon} \ . \tag{3.5}$$

The most common model is the MFA where the linear mapping function of the unobservable factors $\mathbf{Z}$ is defined as

$$f(\mathbf{Z}; \mathbf{A}) = \mathbf{A}_i \mathbf{z}_{ij} + \boldsymbol{\mu}_i \ . \tag{3.6}$$

Here $\mathbf{A}_i$ is a $p \times q$ matrix contains the factor loadings, the latent factors $\mathbf{z}_{ij}$ are defined to be independent and Gaussian with unit variance $\mathcal{N}(0, \mathbf{I}_q)$, and ϵ is random noise distributed independently under $\mathcal{N}(0, \boldsymbol{\Psi}_i)$. Moreover, $\boldsymbol{\Psi}_i$ is a $q \times q$ positive definite symmetric matrix $(i = 1, \ldots, g)$. $\boldsymbol{\mu}_i$ is the non-zero mean vector of the observation data.

With the additional restrictions in Eq. (3.4), the general linear function of MCFA is obtained

$$\begin{aligned} &\mathbf{y}_j = \mathbf{A}\mathbf{z}_{ij} + \boldsymbol{\epsilon}_{ij} \ , \\ &\mathbf{z}_{ij} \sim \mathcal{N}(\boldsymbol{\xi}_i, \boldsymbol{\Omega}_i) \ , \quad \boldsymbol{\epsilon_{ij}} \sim \mathcal{N}(0, \boldsymbol{\Psi}) \ . \end{aligned} \tag{3.7}$$

Here the common loading $\mathbf{A}$ can easily be seen as the transformation matrix, reducing p-dimensional to a latent q-dimensional space. Conventionally, the (unobservable) factors $\mathbf{z}_{ij}$ are distributed independently under $\mathcal{N}(\boldsymbol{\xi}_i, \boldsymbol{\Omega}_i)$, $\boldsymbol{\epsilon}_{ij}$ is random noise distributed independently under $\mathcal{N}(0, \boldsymbol{\Psi})$, and $\boldsymbol{\Psi}$ is a diagonal matrix. Different from MFA, the independent noise variance matrix $\boldsymbol{\Psi}$ is a global parameter instead of the local parameter $\boldsymbol{\Psi}_i$.

With the above definitions, we obtain the conditional distribution of $\mathbf{y}$ in the form

$$P(\mathbf{y}_j \mid \mathbf{z}_{ij}) = \mathcal{N}(\mathbf{y}_i \mid \mathbf{A}\mathbf{z}_{ij}, \boldsymbol{\Psi}) \ . \tag{3.8}$$

Hence, the overall model distribution with the corresponding mixing proportion π_i, $(i = 1, \ldots, g)$ takes the form

$$\begin{aligned} P(\mathbf{y}; \boldsymbol{\theta}) &= \sum_{i=1}^{g} \pi_i \prod_{j=1}^{n} \mathcal{N}(\mathbf{y}_j; \boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2) \\ &= \sum_{i=1}^{g} \pi_i \prod_{j=1}^{n} \mathcal{N}(\mathbf{y}_j; \mathbf{A}\boldsymbol{\xi}_i, \mathbf{A}\boldsymbol{\Omega}_i\mathbf{A}^T + \boldsymbol{\Psi}) \ . \end{aligned} \tag{3.9}$$

Each marginal distribution $\mathcal{N}(\mathbf{y}_j; \mathbf{A}\boldsymbol{\xi}_i, \mathbf{A}\boldsymbol{\xi}_i\mathbf{A}^T + \boldsymbol{\Psi})$ known as a component of this model. By Eq. (3.3), the posterior probability of the observed $\mathbf{y}_j$ belongs to the i^{th} component may be calculated

$$\boldsymbol{\tau}_i(\mathbf{y}_j; \boldsymbol{\theta}) = \frac{\pi_i \boldsymbol{\phi}(\mathbf{y}_j; \mathbf{A}\boldsymbol{\xi}_i, \mathbf{A}\boldsymbol{\xi}_i\mathbf{A}^T + \boldsymbol{\Psi})}{\sum_{h=1}^{g} \pi_h \boldsymbol{\phi}(\mathbf{y}_j; \mathbf{A}\boldsymbol{\xi}_h, \mathbf{A}\boldsymbol{\xi}_h\mathbf{A}^T + \boldsymbol{\Psi})} \ . \tag{3.10}$$

Then the maximum-likelihood technique can be exploited to estimate the parameters and then approximate Maximum A Posteriori by Eq. (3.10). The likelihood function of observed data under MCFA can then be written as

$$\mathcal{L}(\mathbf{y}) = \prod_{i=1}^{g}\prod_{j=1}^{n} P(\mathbf{y}_j \mid \mathbf{z}_{ij}, \omega_i)P(\mathbf{z}_{ij} \mid \omega_i)P(\omega_i) \ . \tag{3.11}$$

Here $\boldsymbol{\omega}$ is the component-indicator labels of this mixture model of g components, whose value is either one or zero depending on whether or not $\mathbf{y}_j$ belongs to the i-th component of the model. Since factors are distributed independently $\mathcal{N}(\boldsymbol{\xi}_i, \boldsymbol{\Omega}_i)$, we have $P(\mathbf{z}_{ij} \mid \omega_i) = \mathcal{N}(\mathbf{z}_{ij} \mid \boldsymbol{\xi}_i, \boldsymbol{\Omega}_i)$. Then, the log-likelihood function is given by

$$\log \mathcal{L}_c(\boldsymbol{\theta}) = \sum_{i=1}^{g}\sum_{j=1}^{n} \omega_i \big\{ \log \pi_i + \log \mathcal{N}(\mathbf{y}_j; \mathbf{A}\mathbf{z}_{ij}, \boldsymbol{\Psi}) + \log \mathcal{N}(\mathbf{z}_{ij}; \boldsymbol{\xi}_i, \boldsymbol{\Omega}_i) \big\} \ . \tag{3.12}$$

In the next subsection, we will introduce how to utilise EM to find the dimensionality reduction matrix $\mathbf{A}$ as well as the parameters of GMM.

Optimisation

The parameters of MCFA can be estimated by the famous EM algorithm, or in particularly, the alternating expectation-conditional maximisation algorithm (AECM) [53, 99]. Then the two-stage form of EM can be shown.

E-step Starting with the initial values for the parameters, we first need to compute the posterior probabilities in Eq. (3.10). The conditional expectation of the component labels $\omega_i (i = 1, \dots, g)$ can be written as $\mathbb{E}_{\boldsymbol{\theta}}\{\omega_i \mid \mathbf{y}_j\} = Pr_{\boldsymbol{\theta}}\{\omega_i = 1 \mid \mathbf{y}_j\} = \boldsymbol{\tau}_i(\mathbf{y}_j; \boldsymbol{\theta})$. Then the expectations of the hidden variables $\mathbb{E}(\mathbf{Z} \mid \mathbf{y}_j, \omega_i)$ and $\mathbb{E}(\mathbf{Z}\mathbf{Z}' \mid \mathbf{y}_j, \omega_i)$ should be estimated that appear in the log-likelihood for all data point $j = 1, \dots, n$ and mixture components $i = 1, \dots, g$. It is easily verified that

$$\mathbb{E}(\mathbf{Z} \mid \mathbf{y}_j, \omega_i) = \boldsymbol{\Omega}_i + \gamma_i^T(\mathbf{y}_j - \mathbf{A}\boldsymbol{\xi}_i) \ , \tag{3.13}$$

$$\mathbb{E}(\mathbf{Z}\mathbf{Z}^T \mid \mathbf{y}_j, \omega_i) = (\mathbf{I}_q - \boldsymbol{\gamma}_i^T A)\boldsymbol{\Omega}_i + \boldsymbol{\theta}^{(k)}(\mathbf{Z} \mid \mathbf{y}_j, \omega_i)\boldsymbol{\theta}^{(k)}(\mathbf{Z} \mid \mathbf{y}_j, \omega_i)^T \ , \tag{3.14}$$

where $\boldsymbol{\gamma}_i = (\mathbf{A}\boldsymbol{\Omega}_i\mathbf{A}^T)^{-1}\mathbf{A}\boldsymbol{\Omega}_i$.

At each iteration, the $\mathbf{Q}(\boldsymbol{\theta}; \boldsymbol{\theta}^{(k)})$ denotes the conditional expectation of Eq. (3.12). Given the observed data $\mathbf{y}_i$ and $\boldsymbol{\theta}^{(k)}$, we have

$$\mathbf{Q}(\boldsymbol{\theta}; \boldsymbol{\theta}^{(k)}) := P(\mathbf{z}^k \mid \mathbf{y}^k; \boldsymbol{\theta}) \ . \tag{3.15}$$

With respect to the posterior distrubutions of both $\mathbf{z}_{ij}$ and ω_i, Eq. (3.15) can be transformed as

$$\begin{aligned} Q(\boldsymbol{\theta};\boldsymbol{\theta}^{(k)}) = \sum_{i=1}^{g}\sum_{j=1}^{n}\boldsymbol{\tau}_{ij}^{(k)}\big\{ \log\pi_i + \mathbb{E}_{\boldsymbol{\theta}^{(k)}}\big\{ \log\mathcal{N}(\mathbf{y}_j;\mathbf{A}\mathbf{z}_{ij},\boldsymbol{\Psi})|\mathbf{y}_j,\omega_i=1\big\} \\ + \mathbb{E}_{\boldsymbol{\theta}^{(k)}}\big\{ \log\mathcal{N}(\mathbf{z}_{ij};\boldsymbol{\xi}_i,\boldsymbol{\Omega}_i)|\mathbf{y}_j,\omega_i=1\big\}\big\} \ . \end{aligned} \tag{3.16}$$

where $\boldsymbol{\tau}_{ij}^{(k)} = \boldsymbol{\tau}_i(\mathbf{y}_j;\boldsymbol{\theta}^{(k)})$. This expectation omitte terms independent of model parameters.

M-step This step involves maximising Eq. (3.16) with the values for parameters $\pi_i^{(k+1)}$, $\boldsymbol{\xi}_i^{(k+1)}$, $\boldsymbol{\Omega}_i^{(k+1)}$, $\mathbf{A}^{(k+1)}$ and $\boldsymbol{\Psi}^{(k+1)}$ by iterate the EM algorithm $(k+1)$ times. The updated estimates of the mixing proportions π_i are derived in the case of the normal mixture model by

$$\pi_i^{(k+1)} = \frac{1}{n}\sum_{j=1}^{n}\boldsymbol{\tau}_{ij}^{(k)} \ , \qquad i=1,\ldots.g \ . \tag{3.17}$$

To consider the other parameters, the local maximisers of log-likelihood function can be obtained by calculating the root of the derivative of Eq. (3.16) with respect to the remaining parameters. The update estimate function of the factor loadings is formed as

$$\mathbf{A}^{(k+1)} = \left(\sum_{i=1}^{g}\mathbf{A}_{1i}^{(k)}\right)\left(\sum_{i=1}^{g}\mathbf{A}_{2i}^{(k)}\right)^{-1} \ , \tag{3.18}$$

where

$$\begin{aligned} \mathbf{A}_{1i}^{(k)} &= \textstyle\sum_{i=1}^{g}\boldsymbol{\tau}_{ij}^{(k)}\{\mathbf{y}_j\mathbb{E}^{(k)}(\mathbf{Z}\mid\mathbf{y}_j,\omega_i^{(k)})\} \ ; \\ \mathbf{A}_{2i}^{(k)} &= \textstyle\sum_{i=1}^{g}\boldsymbol{\tau}_{ij}^{(k)}\{\mathbb{E}^{(k)}(\mathbf{Z}\mathbf{Z}'\mid\mathbf{y}_j,\omega_i^{(k)})\} \ . \end{aligned}$$

The $\boldsymbol{\xi}_i^{(k+1)}$ and $\boldsymbol{\Omega}_i^{(k+1)}$ can be expressed as follows

$$\boldsymbol{\xi}_i^{(k+1)} = \boldsymbol{\xi}_i^{(k)} + \frac{\sum_{j=1}^{n}\boldsymbol{\tau}_{ij}^{(k)}\beta^{(k)^T}\boldsymbol{\Sigma}_i^{(k)^T}\boldsymbol{\mu}_{ij}}{\sum_{j=1}^{n}\boldsymbol{\tau}_{ij}^{(k)}} \ , \tag{3.19}$$

$$\boldsymbol{\Omega}_i^{(k+1)} = \frac{\sum_{j=1}^{n}\boldsymbol{\tau}_{ij}^{(k)}\boldsymbol{\Sigma}_i^{(k)^T}\beta^{(k)^T}\boldsymbol{\mu}_{ij}^T\boldsymbol{\Sigma}_i^{(k)}\beta^{(k)}}{\sum_{j=1}^{n}\boldsymbol{\tau}_{ij}^{(k)}} + (\mathbf{I}_q - \boldsymbol{\Sigma}_i^{(k)}\beta^{(k)}\mathbf{A}^{(k)})\boldsymbol{\Omega}_i^{(k)} \ , \tag{3.20}$$

$$\beta^{(k)} = \mathbf{A}^{(k)}\boldsymbol{\Omega}_i^{(k)}, \quad \boldsymbol{\mu}_{ij} = \mathbf{y}_j - \mathbf{A}^{(k)}\boldsymbol{\xi}_i^{(k)} \ ,$$
$$\boldsymbol{\Sigma}_i^{(k)} = \left(\mathbf{A}^{(k)}\boldsymbol{\Omega}_i^{(k)}\mathbf{A}^{(k)^T} + \boldsymbol{\Psi}^{(k)}\right)^{-1} .$$

The updated estimates $\mathbf{\Psi}^{(k+1)} = diag\big(\mathbf{\Psi}_1^{(k)} + \mathbf{\Psi}_2^{(k)}\big)$, where

$$\mathbf{\Psi}_1^{(k)} = \frac{\sum_{i=1}^{g}\sum_{j=1}^{n} \boldsymbol{\tau}_{ij}^{(k)} \mathbf{\Psi}^{(k)} (\mathbf{I}_p - \mathbf{\Sigma}_i^{(k)} \mathbf{\Psi}^{(k)})}{\sum_{i=1}^{g}\sum_{j=1}^{n} \boldsymbol{\tau}_{ij}^{(k)}},$$

$$\mathbf{\Psi}_2^{(k)} = \frac{\sum_{i=1}^{g}\sum_{j=1}^{n} \boldsymbol{\tau}_{ij}^{(k)} \mathbf{\Psi}^{(k)^T} \mathbf{\Sigma}_i^{(k)^T} \boldsymbol{\mu}_{ij} \boldsymbol{\mu}_{ij}^T \mathbf{\Sigma}_i^{(k)} \mathbf{\Psi}^{(k)}}{\sum_{i=1}^{g}\sum_{j=1}^{n} \boldsymbol{\tau}_{ij}^{(k)}}.$$

3.1.3 Experiments

In this section, we conduct a series of experiments on one simulated and four real datasets (obtained from UCI machine learning repository [103]) and compare the performance of the joint learning approach MCFA with the independent learning method PCA followed by GMM and another competitive joint leaning method mPPCA [41, 43], which is stated in Chapter 3.3.1.

Experimental Setup: All the evaluations are conducted based on clustering, since we merely consider unsupervised dimensionality reduction in this section. All the data are applied as training so as to get an appropriate clustering result. Following previous research, we report the error rate (ERR), the adjust rand index (ARI), and the Bayesian information criterion (BIC) to compare different algorithms.[1] Note that, although we do not exert any labeled information in clustering, the label for each sample is known beforehand in all the used datasets. Hence we can exploit the label information to evaluate the clustering performance. Specifically, we could exploit ERR as the evaluation metric for clustering, since we expect the same cluster should contain the data of the same class. BIC is employed to determine the number of clusters where the lower BIC implies better fitting [104]. ARI is a measure of similarity between two clusterings. If the clustering and the true class perfectly agree, the value of the ARI will be 1 [105].

Simulation Data

To validate the effectiveness of the joint learning approach MCFA, we first performed a simulation experiment. We generated 300 random vectors from each of $g = 3$ different three-dimensional multivariate normal distributions. The three distributions have respec-

[1] $AdjustedIndex = \frac{Index - ExpectedIndex}{MaxIndex - ExpectedIndex}$;
$BIC = -2 \times \ln(Likelihood) + k \times \ln($the number of samples$)$.

Table 3.1: Comparison among the MCFA, PCA+GMM and mPPCA on Simulated Data

		MCFA			PCA+GMM [41]			mPPCA [43]		
CLUS	DIM	ERR	BIC	ARI	ERR	BIC	ARI	ERR	BIC	ARI
2	2	0.3333	4173	0.5600	0.3333	3153	0.5553	0.3333	**62171**	0.4831
3	2	**0.0100**	**4105**	**0.9702**	**0.0300**	**3080**	**0.9126**	**0.0233**	88344	**0.9310**

* CLUS denotes the number of clusters; DIM denotes the number of dimensions.

tively means $\mu_1 = (0,0,0)^T, \mu_2 = (2,2,6)^T, \mu_3 = (8,8,8)^T$, and covariance matrices

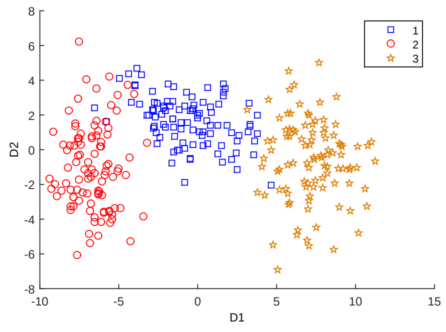
$$\Sigma_1 = \begin{pmatrix} 4 & -1.8 & -1 \\ -1.8 & 2 & 0.9 \\ -1 & 0.9 & 2 \end{pmatrix}, \Sigma_2 = \begin{pmatrix} 4 & 1.8 & 0.8 \\ 1.8 & 2 & 0.5 \\ 0.8 & 0.5 & 2 \end{pmatrix}, \Sigma_3 = \begin{pmatrix} 4 & 0 & -1 \\ -1.8 & 2 & 0.9 \\ -1 & 0.9 & 2 \end{pmatrix}.$$

To compare the performance MCFA with PCA for DR. We plot the unsupervised feature reduction results on Fig. 3.1. In Fig. 3.1(a), the optimal subspace is just obtained by reducing the observations with PCA. Clearly, the original clustering information among data was less obvious after PCA. For MCFA, the predicted observations in the optimal subspace shown in Fig. 3.1(b) is given by $\hat{\mathbf{z}}_{\mathbf{ij}} = \mathbf{A}\sum_{i=1}^{g} \boldsymbol{\tau}_{ij}\mathbf{y}_{ij}$. Since mPPCA assume to project the data of different clusters into different subspaces, the mPPCA does not show all the low-dimensional representation in a subspace. Obviously, the joint learning approaches could lead to much better clustering performance, especially compared with PCA.

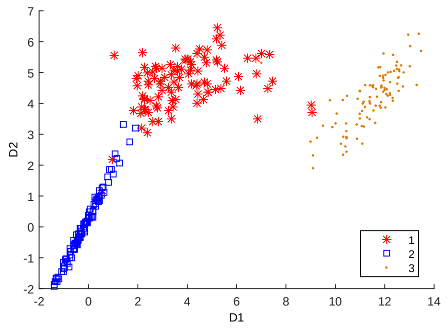
To quantitatively evaluate the clustering performance, we compute ERR, ARI and BIC with PCA followed by GMM and the joint learning MCFA and mPPCA. These results are shown in Table 3.1. From the table, the lowest BIC of both approaches are pointed to 3 clusters, indicating that 3 is the best cluster number. Moreover, in the case of 3 cluster number, the joint learning MCFA outperformed PCA followed by GMM significantly in terms of the other two criteria.

Comparison on Real Data

Datasets Table 3.2 summarises the real-world datasets used in our experiments (details are shown in appendix). We have compared the joint learning MCFA, mPPCA and independent learning approach PCA followed by GMM in case of various cluster number and different dimensionality ranged from 2 to the feature number, as the cluster number was usually not known. For iris data, one class is linearly separable from the other two, and



(a) 2-d subspace by PCA



(b) 2-d subspace by joint learning

Fig. 3.1: Comparison of DR by PCA and joint learning models on simulated data. Data points with the same shape are supposed to be clustered together.

the latter are not linearly separable from each other. We also used seeds dataset to study methods for clustering in high-dimensional data with large noise information. Each of the subjects consists of a total of 37 variables ($p = 37$), including 7 geometric parameters of wheat kernels and 30 uniform noise variables. The noise variables were generated from the uniform distribution on the interval $[0, 2]$.

Table 3.2: Summary of real-world datasets

Datasets	Size ($N \times P$)	Details
User knowledge model	403×5	4 knowledge levels of the students
Physical	178×13	The chemical analysis of wines with 3 cultivars
Iris Data	150×4	3 classes of iris plant.
Seeds Data	210×37	3 different varieties of wheat

Experimental results The results of the user knowledge modelling data are shown in Tabel 3.3, the best-estimated cluster number of MCFA is 4 according to the lowest BIC. This setting also achieved the lowest ERR, and the highest ARI when reducing the dimensional to 2 factors. In the same way, with the lowest BIC, the result by mPPCA is 3 clusters and 2 clusters given by PCA. All of these results do not conform to the ground truth. To show the comparison better, we also plot the results in Fig. 3.2. It is clearly observed that MCFA is also significantly better than the best case of other competitive methods by comparing ERR.[2]

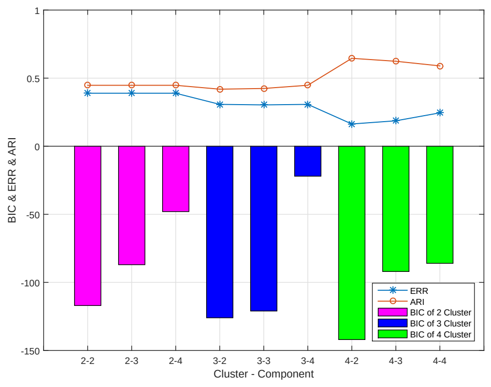
[2]When the estimate of the variance of the random error term is too small, the value of log-likelihood can be a positive number. This is the reason why BIC becomes negative. If ERRs/ARIs are same, the model with the lower BICs are better.

Table 3.3: Comparison among MCFA, PCA+GMM and mPPCA on User Knowledge Data

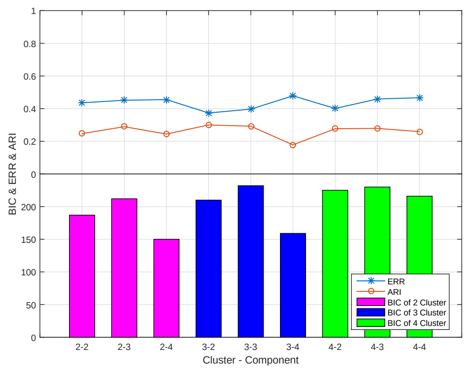
User Knowledge Modelling Data

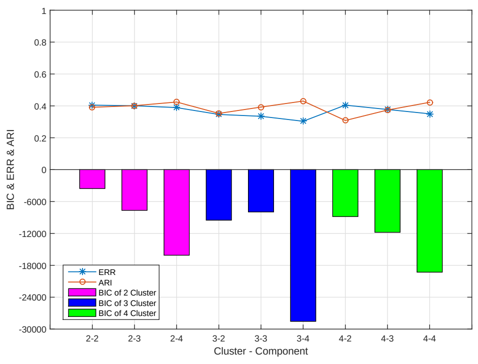
		MCFA			PCA+GMM			mPPCA		
CLUS	DIM	ERR	BIC	ARI	ERR	BIC	ARI	ERR	BIC	ARI
2	2	0.3891	−117	0.4474	0.4358	187	0.2469	0.4047	−3554	0.3912
	3	0.3891	−87	0.4474	0.4514	212	0.2896	0.4008	−7658	0.4013
	4	0.3891	−48	0.4474	0.4553	**150**	0.2442	0.3891	−16103	0.4242
3	2	0.3074	−126	0.4190	**0.3735**	210	**0.3001**	0.3463	−9509	0.3528
	3	0.3035	−121	0.4242	0.3969	232	0.2924	0.3346	−7956	0.3926
	4	0.3074	−22	0.4477	0.4786	159	0.1781	**0.3035**	**−28548**	**0.4302**
4	2	**0.1634**	**−142**	**0.6456**	0.4008	225	0.2771	0.4047	−8825	0.3084
	3	0.1868	−92	0.6240	0.4591	230	0.2791	0.3774	−8158	0.3739
	4	0.2451	−86	0.5901	0.4669	216	0.2593	0.3502	−23606	0.4224

* CLUS denotes the number of clusters; DIM denotes the number of dimensions.



(a) MCFA





(b) PCA+GMM (c) mPPCA

Fig. 3.2: Three criteria comparison on User Knowledge Dataset among different algorithms. Lines represent ERR, ARI and histograms represent BIC where different colours represent different numbers of clusters.

Furthermore, in Table. 3.4, the best-estimated cluster number of MCFA and PCA+GMM is 3, according to the lowest BIC of Physical Data. Obviously, in Fig. 3.3, in the case of $q \geq 6$ factors with 3 clusters, the joint learning led to better performance than PCA+GMM in terms of ERR and ARI. However, with the mPPCA approach, it did not lead to the s-mallest error rate with the lowest BIC.

For iris data, all algorithms achieved the lowest BIC with 3-cluster (see Table 3.5). This also matches the class number in this dataset. Since two of classes are not linearly separable, the error rates are all 0.3333 when cluster number is set to 2. In Fig. 3.4, MCFA achieved the lowest ERR and the highest ARI, which outperformed significantly that of other competitors.

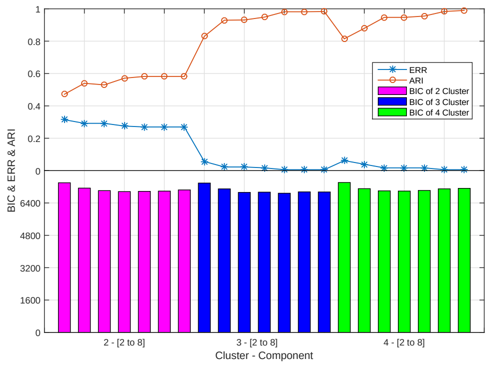
Table 3.6 presents the seeds dataset which is high-dimensional data with large noise information. By comparing BIC, 3 clusters are the best for MCFA and mPPCA, and 2 clusters are the best for PCA. It can be seen clearly in Fig. 3.5, the MCFA leads to a

Table 3.4: Comparison among MCFA, PCA+GMM and mPPCA on Physical Data

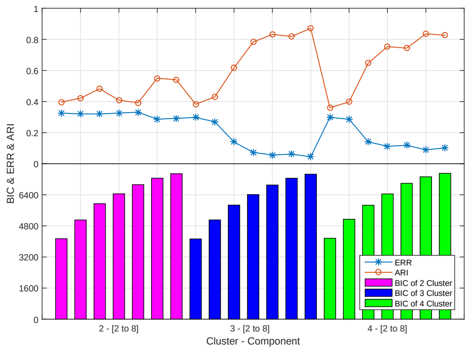
Physical Data

		MCFA			PCA+GMM			mPPCA		
CLUS	DIM	ERR	BIC	ARI	ERR	BIC	ARI	ERR	BIC	ARI
2	2	0.3146	7398	0.4717	0.3258	4142	0.3963	0.3090	241520	0.4336
	3	0.2921	7134	0.5397	0.3202	5106	0.4219	0.3090	240639	0.4403
	4	0.2921	7010	0.5298	0.3202	5945	0.4820	0.3315	250914	0.3904
	5	0.2753	6962	0.5711	0.3258	6452	0.4088	0.3315	250091	0.3904
	6	0.2697	6973	0.5820	0.3315	6922	0.3916	0.3315	250283	0.3904
	7	0.2697	6986	0.5820	0.2865	7255	0.5499	0.3315	**236383**	0.3904
	8	0.2697	7045	0.5820	0.2921	7487	0.5397	0.3315	263761	0.3904
3	2	0.0562	7384	0.8298	0.2978	**4130**	0.3827	0.2191	527095	0.5053
	3	0.0225	7096	0.9295	0.2697	5109	0.4302	0.1067	626431	0.7174
	4	0.0225	6922	0.9309	0.1401	5872	0.6170	0.0787	688774	0.7781
	5	0.0169	6935	0.9485	0.0730	6413	0.7822	0.1461	678949	0.6381
	6	**0.0056**	**6881**	0.9832	0.0562	6905	0.8319	0.0899	683243	0.7486
	7	0.0056	6948	0.9817	0.0618	7253	0.8185	**0.0674**	672007	**0.8069**
	8	0.0056	6944	0.9832	**0.0449**	7462	**0.8708**	0.0843	594161	0.7637
4	2	0.0618	7411	0.8145	0.2978	4165	0.3600	0.2640	538754	0.4802
	3	0.0393	7106	0.8792	0.2865	5143	0.3977	0.2416	573750	0.5014
	4	0.0169	6999	0.9470	0.1404	5863	0.6479	0.1854	677267	0.6478
	5	0.0169	6988	0.9470	0.1124	6445	0.7531	0.2191	656419	0.5858
	6	0.0169	7018	0.9551	0.1180	6992	0.7436	0.1966	647137	0.6339
	7	0.0056	7099	0.9833	0.0899	7327	0.8355	0.2472	549149	0.5043
	8	0.0056	7121	**0.9900**	0.1011	7505	0.8264	0.2165	682489	0.5709

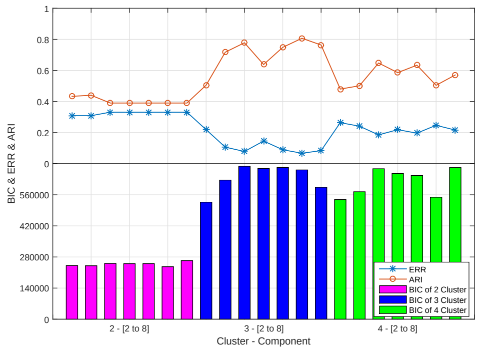
* CLUS denotes the number of clusters; DIM denotes the number of dimensions.



(a) MCFA



(b) PCA+GMM



(c) mPPCA

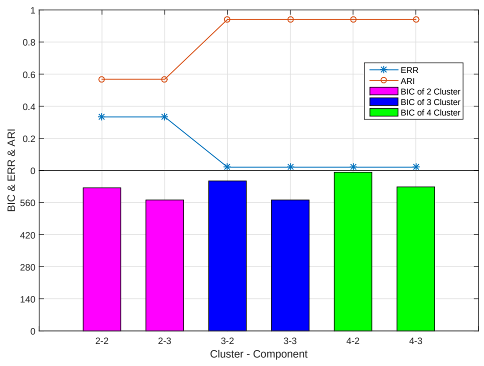
Fig. 3.3: Three criteria comparison on user Physical Dataset among different algorithms. Lines represent ERR, ARI and histograms represent BIC where different colours represent different numbers of clusters.

Table 3.5: Comparison among MCFA, PCA+GMM and mPPCA on Iris Data

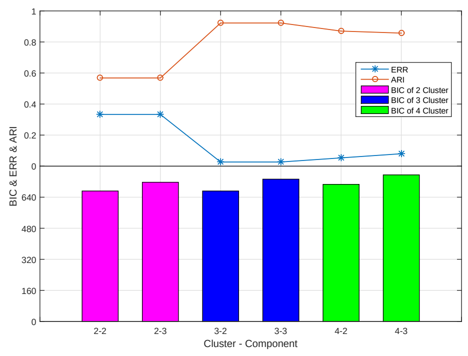
Iris Data

		MCFA			PCA+GMM			mPPCA		
CLUS	DIM	ERR	BIC	ARI	ERR	BIC	ARI	ERR	BIC	ARI
2	2	0.3333	624	0.5681	0.3333	672	0.5681	0.3333	50768	0.5681
	3	0.3333	571	0.5681	0.3333	717	0.5681	0.3333	34002	0.5681
3	2	0.0200	654	0.9410	**0.0267**	**672**	**0.9222**	0.0267	73706	0.9222
	3	**0.0200**	**571**	**0.9410**	0.0267	733	0.9222	**0.0267**	**32758**	**0.9222**
4	2	0.0200	692	0.9410	0.0533	706	0.8700	0.0600	57953	0.8488
	3	0.0200	628	0.9410	0.0800	755	0.8570	0.0400	44837	0.9116

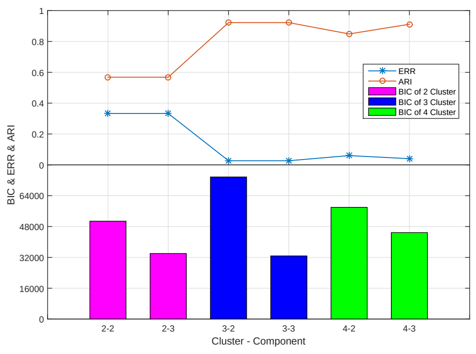
* CLUS denotes the number of clusters; DIM denotes the number of dimensions.



(a) MCFA



(b) PCA+GMM



(c) mPPCA

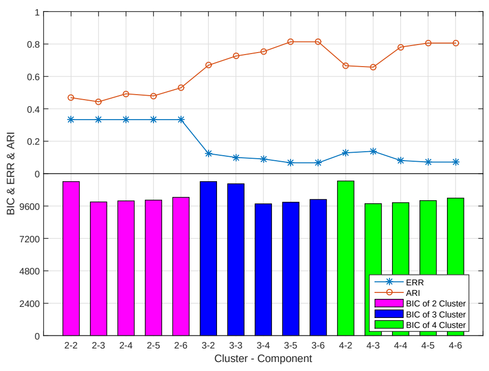
Fig. 3.4: Three criteria comparison on Iris Dataset among different algorithms. Lines represent ERR, ARI and histograms represent BIC where different colours represent different numbers of clusters.

Table 3.6: Comparison among MCFA, PCA-GMM and mPPCA on Seeds Data

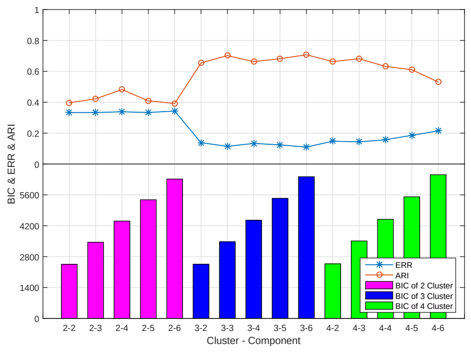
Seeds Data

		MCFA			PCA+GMM			mPPCA		
CLUS	DIM	ERR	BIC	ARI	ERR	BIC	ARI	ERR	BIC	ARI
2	2	0.3333	11410	0.4685	0.3333	**2453**	0.3963	0.3333	**18679**	0.4720
	3	0.3333	9902	0.4436	0.3333	3453	0.4219	0.3333	21156	0.4517
	4	0.3333	9980	0.4927	0.3381	4413	0.4820	0.3333	20932	0.4564
	5	0.3333	10040	0.4797	0.3333	5378	0.4088	0.3333	22514	0.4564
	6	0.3333	10242	0.5299	0.3429	6318	0.3916	0.3381	20919	0.4683
3	2	0.1238	11411	0.6688	0.1381	2460	0.6544	**0.1143**	21192	**0.6945**
	3	0.1000	11247	0.7257	0.1143	3473	0.7018	0.1238	25097	0.6807
	4	0.0905	**9763**	0.7530	0.1333	4449	0.6627	0.1333	26369	0.6632
	5	**0.0667**	9883	**0.8138**	0.1238	5444	0.6814	0.1429	28910	0.6452
	6	0.0667	10086	0.8136	**0.1095**	6422	**0.7072**	0.1619	28872	0.6046
4	2	0.1286	11455	0.6660	0.1476	2475	0.6632	0.1714	22517	0.6364
	3	0.1381	9776	0.6572	0.1429	3510	0.6812	0.1857	27477	0.6358
	4	0.0810	9849	0.7811	0.1571	4491	0.6319	0.2095	27424	0.5915
	5	0.0714	10002	0.8059	0.1857	5510	0.6102	0.2190	27411	0.6271
	6	0.0714	10188	0.8059	0.2143	6512	0.5310	0.2381	28528	0.5782

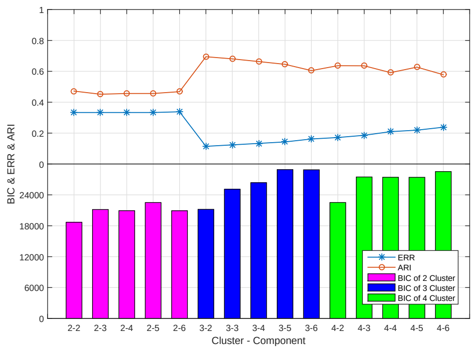
* CLUS denotes the number of clusters; DIM denotes the number of dimensions.



(a) MCFA



(b) PCA+GMM



(c) mPPCA

Fig. 3.5: Three criteria comparison on Seeds Dataset among different algorithms. Lines represent ERR, ARI and histograms represent BIC where different colours represent different numbers of clusters.

good result of error rate for the cases that the number of factors is greater than 3. The highest ARI is obtained by using five factors ($q = 5$) in MCFA. The best result of ERR and ARI for mPPCA is obtained by reducing to two dimensions. However, it dose not lead to the choice of q with the best result when BIC is exploited to choose g with PCA. To comparing the best result of two approaches, the ERR and ARI for MCFA are much better than others.

In summary, all the experimental results show that the joint learning model significantly outperforms the independent learning models. In the next section, we will compare two classic joint learning models.

3.2 Joint Learning

The latent variable models with density estimation are the fundamental machine learning approaches which have achieved big success in both supervised learning, e.g., classification and regression, and unsupervised learning, e.g., clustering. When latent variable models (e.g. GMM) are employed for practical data, they are usually to perform dimensionality reduction beforehand. The purpose is both to reduce the computational time for high dimensional data and to find a suitable subspace where better clustering or classification performance could be achieved due to the removal of possible noisy features. In this setting, the optimal subspace and the following optimal parameters of GMM are searched separately or independently. The optimal subspace obtained by the independent DR may not be appropriate for the following GMM. This is particularly the case in the context of unsupervised learning, e.g., clustering. In supervised learning, class labels could be used for deriving a good subspace, while in unsupervised learning, the principles used for DR (e.g., maximisation of the variance in PCA) may not be appropriate for GMM [8].

To handle unsupervised dimensionality reduction for GMM, we argue that both the optimal subspace and the parameters for GMM should be jointly learned. This is significantly different from the traditional setting that the two steps are usually conducted separately. As a classical joint learning method, MFA has introduced the factor loading matrices as an additional latent variable for a mixture model (shown in Chapter 2.2.3). Each factor loading matrix can be thought of as a dimensionality reduction matrix. Therefore, the MFA model performs the dimensionality reduction of data by making locally linear assumptions, which projects the points of different clusters into different subspaces. Fig. 3.6 illustrates a joint learning method with locally linear assumptions to perform the dimensionality reduction and clustering simultaneously. The principle is to maximise the homogeneity of points from the same cluster.

Based on extending the classical model, it is engaged as a global linear assumption that all latent factors share a common factor loading. In the proposed joint learning model, the common factor loading is treated as a global dimensionality reduction matrix, while the mixture of latent factors can be regarded as a GMM. Specifically, this model could reduce the data dimensions by a single projection matrix, which can project the different clusters into a sub-space, as shown in Fig. 3.7. Different from local linear models, global linear models not only follow the principle of maximising the similarity but also to maximise the distance between clusters. When the information provided by the sample is insufficient (small sample size data), the global matrix can utilise the interaction attributes among clusters to reduce the dimensionality, and it also reduces the number of the parameters to make the model easier to learn.

3.2.1 Supplementary Experiments: MFA vs MCFA

In unsupervised learning, two joint learning models, MFA and MCFA, are able to present arbitrarily complex probability density functions, which fact makes them an excellen-

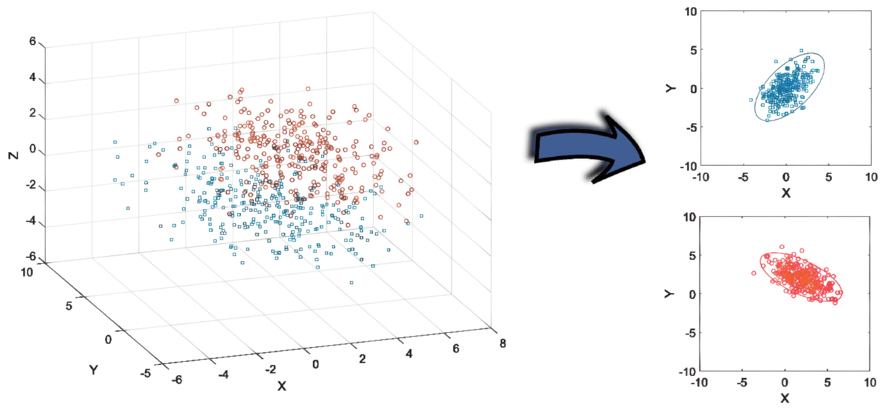


Fig. 3.6: The sketch of clustering and dimensionality reduction. Joint learning with locally linear assumptions drops the data points of different clusters into different sub-spaces and cluster them at the same time. Different colours represent different clusters.

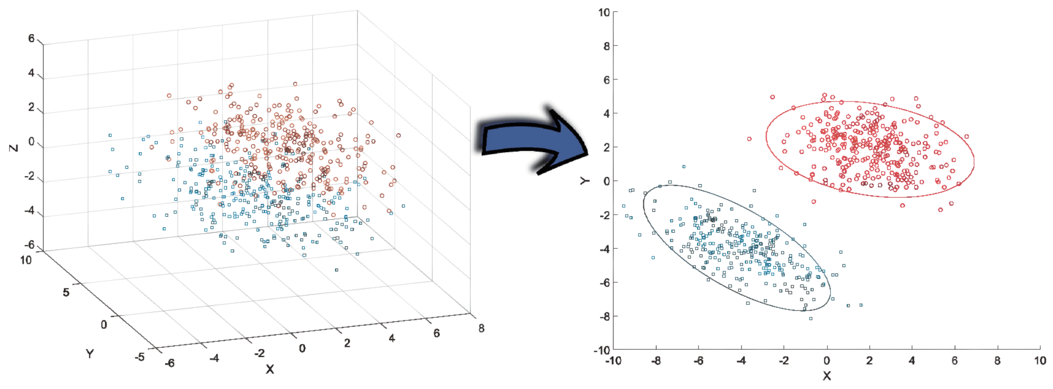


Fig. 3.7: The sketch of clustering and dimensionality reduction. Joint learning with globally linear assumptions drops the data points of different clusters into a common sub-spaces and cluster them simultaneously. Different colours represent different clusters.

t choice for representing complex low-dimensional points [54]. As the supplementary experiments, we conduct complementary experiments on various real-world datasets to evaluate the performance of both joint learning models, including artificial data, grey images, and digitised aerial image. The following datasets are exploited in our empirical experiments.

- ULC-3: The urban land cover (ULC) data are used to classify a high-resolution aerial image which consists of 3 types with 273 training samples, 77 test samples and 147 attributes [106][107].
- Coil-4-proc: This dataset contains images for 4 objects discarding the background and each object has 72 samples [108]. The images are downsampled into 32 by 32 pixels and then reshaped to a 1024-dimensional vector. There are just 248 samples in the training set and 40 samples in the test set.
- Leuk72_3k: This dataset is an artificial dataset including 3 classes which have been drawn from randomly generated Gaussian mixtures. The Leuk72_3k has only 54 training samples and 18 test samples with 39 attributes.
- USPS1-4: This handwriting digit data contains 1 to 4 digits images of size 16 by 16 pixels. Each image is reshaped to a 256-dimensional vector. The training set includes 100 samples of each digit and the test set also consists of 100 of each digit.

It is worth noting that data of the training set are used for training a density model and clustering the data points without any label information. As the model-based method on clustering, the testing set is used to test the adaptability of the model on more data form homogenous distributed.

Empirical Results

Empirically, as a criterion, the average log-likelihood is used for examining the quality of the density estimates after modelling the density of the observed data. The empirical results are demonstrated on both MFA and MCFA. The model parameters are estimated over the training data by maximising the log-likelihood value. By multiple trials, the average log-likelihood value on the training data are shown in Fig. 3.8. In order to intuitively observe the trend of the entire experimental results, the image results have to be treated as logarithmic, which is utilised to control all the results in the same range. On the testing data, the log-likelihood value obtained using only the model has been trained and did not update the model parameters, and the results are shown in Fig. 3.9. From the comparison of both results, the log-likelihood values obtained by the MCFA model are higher than that of MFA on all datasets. Consequently, sharing a common loading can improve the true log-likelihood dramatically.

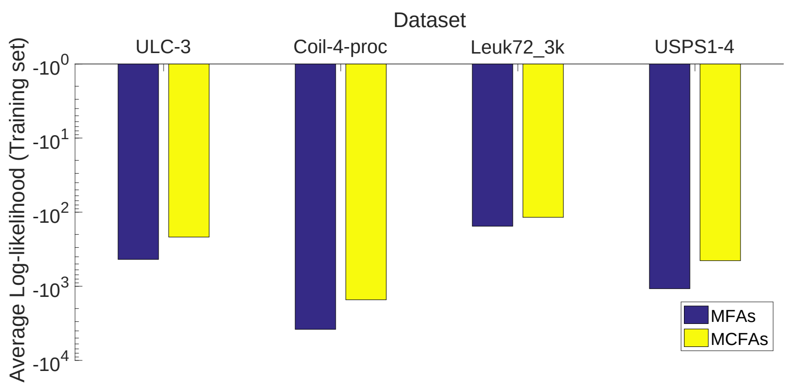


Fig. 3.8: Performance on various real data (on the training set) in terms of the log-likelihood (the larger, the better).

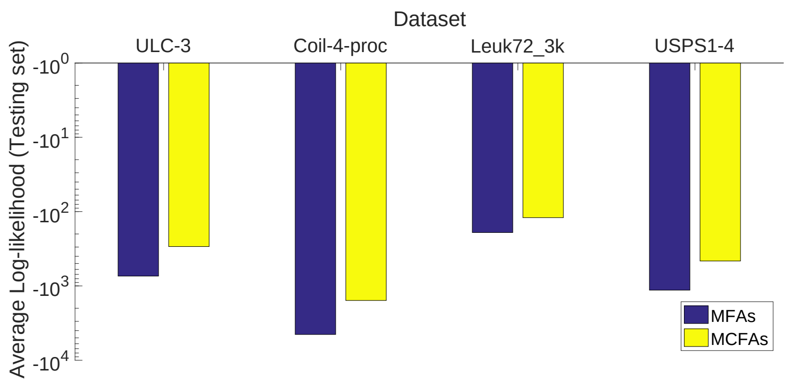


Fig. 3.9: Performance on various real data (on the testing set) in terms of the log-likelihood (the larger, the better).

Clustering Result

Then, the clustering error rate is demonstrated of both training and testing datasets, and the best results are reported from multiple trials. In the experiments, both methods have been initialised by a random assortment. Also, the number of mixes is set to be the same as the number of real categories. Comparing both MFA and MCFA results on the training data in Fig. 3.10, the results of these two methods are not significantly different on the Coil-4-proc dataset, and even get the same result on the Leuk72_3k dataset. As model-based clustering methods, they have the ability to cluster datasets that are never seen but have the same distribution with the training data. Comparing the results in Fig. 3.11, the

MCFA model still maintains good performance on the test dataset. On the whole, the results of MCFA are consistently better than that of MFA.

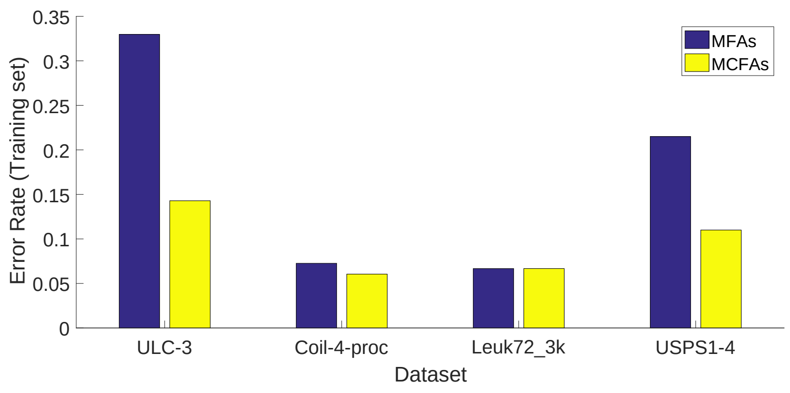


Fig. 3.10: Clustering error rate on 4 datasets. The best result is reported from each model on the training set.

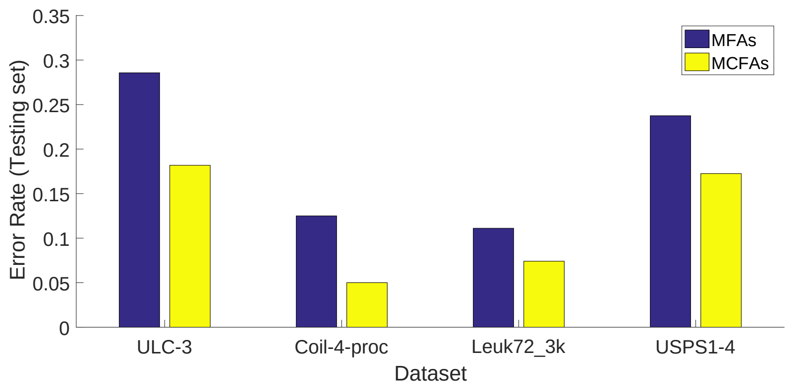


Fig. 3.11: Clustering error rate on 4 datasets. The best result is reported from each model on the testing set.

3.3 Two-layer Mixtures of Factor Analysers with Joint Factor Loading

The generic discrimination problem is considered as the assignment of a previously unseen object to a class of training data data which consist of class labelled measurements (and possibly some unlabelled measurements). When the generative model is involved, its objective becomes to model the measurements belonging to different labels. In the topic of discriminant analysis, mixture models are also very important and have been studied intensely in the relevant literature [109, 110]. When a finite mixture model is established using label information, the measurements for each class are modelled as a distribution that is considered a component of mixtures. Then a Bayesian formalism is developed to obtain the parameters of mixture models and infers the posterior distributions as the prediction model [111].

Traditionally, each distribution of the mixture model represents the data with the same label to performed as a measurement of the discrimination analysis [109, 111]. However, in the case of learning high-dimensional data with complicated data manifolds, distribution is usually not enough to represent a class of data [8]. This is particularly the case for the small sample size (S3) problem [112, 113], where the data patterns are high-dimensional but of low cardinality. In such problems, each class of data cannot well learn a model, and the subspace derived by the independent DR may even significantly deteriorate the discrimination performance.

Motivated from the above issues, we propose a mixture discrimination model within an FA framework, a novel model referred to as the two-layer mixtures of factor analysers with joint factor loading (2L-MJFA). This relies upon a mixture of mixtures structure [3], which is used to capture the complex properties of each class better and realise the joint learning requirements efficiently. An important characteristic of 2L-MJFA is that all of its involved latent factors are designed to share the same loading matrix (shown in Fig. 3.12(c)). This has a dual purpose, in the sense that, on the one hand, it operates as the driving DR structure, and on the other hand it significantly reduces the number of parameters. The latter accelerates training while mitigates the negative effect caused by the limited number of per class samples.

Contrary to the independent approaches, the proposed 2L-MJFA is capable of simultaneously learning the DR matrix as well as the optimal parameters of the classification model. This model is implemented via a GMM for simplicity, but it is straightforward to extend the two-layer mixture approach to the use of other models. Through joint learning, the method achieves efficient DR that not only reduces the computational time for high dimensional data but more importantly it significantly benefits the final classification stage. Another contribution is that we also propose a modified expectation-maximisation (EM) algorithm that consists of two-layer loops so that the joint learning is conducted very efficiently. The first layer loop is used to estimate the joint parameters that fit the mixture among different classes, whereas the second one trains the mixture components within each class. The 2L-MJFA is theoretically distinct to other joint learning FA-based models, such as the MCFA model, the mixtures of MCFAs (mMCFA), and the mixtures of probabilistic PCAs (mPPCA) [27, 43, 100]. Further details about these models are presented in the following section. Our experiments show that the proposed method significantly outperforms these existing methods in seven benchmark datasets.

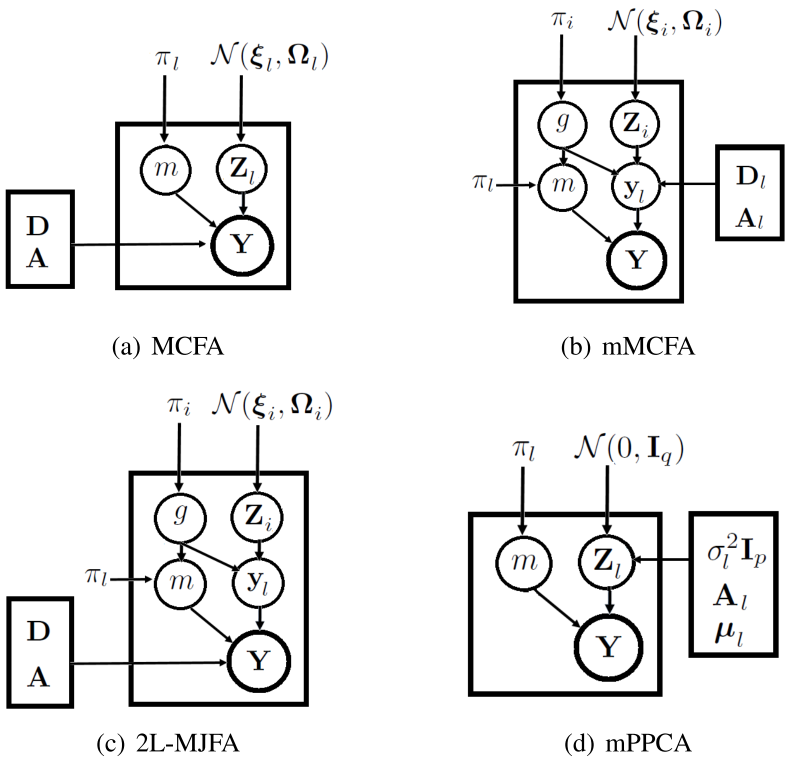
[3]In each class, the mixture components can clearly detail a multimodal density.

The rest of this section is organised as follows. Section 3.3.1 briefly reviews related work and emphasises the differences between our proposed approach and existing ones. The baseline model MCFA is introduced as preliminaries in Section 3.3.2. In Section 3.3.3 we introduce the proposed 2L-MJFA model, while Section 3.3.4 explains how the model parameters can be estimated by the modified EM algorithm. In Section 3.3.5 we present the experimental setup and the classification results with the aid of seven datasets including a synthetic dataset and six real ones. This work has been published on [17] and [19].

3.3.1 Related Model Architectures and Computational Complexities

There have been several joint learning FA-based approaches [114, 115] related to our proposed method. To illustrate the distinction, we present the different alternative structures incorporated in various models in Fig. 3.12. In particular, the MFA model [116] is the base model for what we propose. It combines DR with clustering and utilises a subspace metric to guide cluster separation. This work is extended by MCFA [100] which assumes the factor loading of the MFA model to be a common matrix that can largely reduce the associated parameters. When MCFA is used for classification, one straightforward way is to regard each class as one component, as shown in Fig.3.12(a). Obviously, such a setting is quite basic and not adequately flexible, since data classes may have complex distributions and modalities. Another popular variant that extends MCFA is mMCFA (shown in Fig. 3.12(b)), where the factor loadings $\mathbf{A}_i$ are different for each class. In general, different loading matrices imply independent DR for different classes, and this may be physically impractical. More importantly, mMCFA could be problematic in S3 problems, as the limited number of samples cannot support the accurate learning of the loading matrices. To this end, a non-trivial model is proposed here by sharing one loading matrix for all the classes. The mPPCA method [43] extends PCA to a mixture distribution model. As seen in Fig.3.12(d), its graphical model is quite similar to MFA with the elements of the common covariance matrix $\mathbf{D} = \sigma^2\mathbf{I}_p$ assumed to be isotropic [117], where $\mathbf{I}_p$ is the p-dimensional identity matrix. For classification, each class is modelled as an mPPCA model. This method is limited due to its poor flexibility and has many redundant parameters for dealing with $S3$ problems.

We now analyse the parameter numbers in the different models, assuming p dimensions, q reduced dimensions from p, and m classes. Setting g mixture components in each class, the covariance matrix of each component has $N = \frac{p(p+1)}{2}$ parameters. Since mPPCA converts the diagonal covariance matrix into an isotropic one as $\mathbf{\Sigma}_i = \mathbf{W}_i\mathbf{W}_i^T + \sigma^2\mathbf{I}_p$, where factor loading $\mathbf{W}_i \in \mathbb{R}^{p\times q}$ contains $\frac{q(q-1)}{2}$ constraints, its total



(a) MCFA (b) mMCFA

(c) 2L-MJFA (d) mPPCA

Fig. 3.12: Comparison of different models. $\mathbf{Y}$ denotes observed data, $\mathbf{Z}_l$/$\mathbf{Z}_i$ denotes latent factors, and $\mathbf{A}$/$\mathbf{A}_l$ is (common) factor loadings, where m is the class label and g is the number of mixtures. (a) MCFA which is the fundamental MFA model with a common $\mathbf{A}$. (b) Mixtures of MCFAs with each class consisting of a components mixture with individual local factor loadings $\mathbf{A}_l$. (c) The proposed 2L-MJFA with a global factor loading $\mathbf{A}$ shared between and within classes in the 2-layer mixture model. (d) Mixtures of probabilistic PCA which is similar to MFA but with an isotropic common covariance matrix.

number of parameters is

$$N_1 = m\left(g + gp + gpq - \frac{gq(q-1)}{2}\right) .$$

If either p or q is large, the number of parameters may not even be manageable with a diagonal covariance. To further reduce the parameters and accelerate training, the component covariance matrices of mMCFA has a factor-analytic representation $\boldsymbol{\Sigma}_i = \mathbf{A}\boldsymbol{\Omega}_i\mathbf{A}^T + \mathbf{D}$, where $\mathbf{D}$ is a diagonal matrix, and $\mathbf{A}$ contains the factor loading for all the components [99]. From the orthogonality requirement, $\mathbf{A}$ has $pq - q^2$ constraints. Hence, in mMCFA the total number of parameters is reduced to

$$N_2 = m\left[pq - q^2 + p + g\left(1 + q + \frac{q(q+1)}{2}\right)\right] .$$

Table 3.7 lists the associated parameter numbers for FA models. Since $p >> q$, the order of the number of parameters can be approximated via the simpler form shown in the right-

Table 3.7: Summary of the number of parameters for the main models. The rightmost column shows the simplified number of parameters which makes the total numbers easy to compare.

Model:	Number of parameters:	Approximation:
mPPCA	$m[g + gp + gpq - \frac{gq(q-1)}{2}]$	$(mg + mq)p$
mMCFA	$m[pq - q^2 + p + g(1 + q + \frac{q(q+1)}{2})]$	$(m + mq)p$
2L-MJFA	$pq - q^2 + p + m[g + gq + \frac{gq(q+1)}{2}]$	$(q + 1)p$

most table column. It can be seen that the proposed 2L-MJFA requires the least number of parameters, which ultimately make it more suitable for dealing with S3 problems; this is also verified by the experimental results.

3.3.2 Problem Definition

As a multi-linear model, Mixtures of Factor Analysers via a common loading decomposes a common factor loading to cross linear subspaces within the covariate vector space, making factors have lower dimension than the covariates. The notation in this section will largely follow Section 3.1.2. Let $\mathbf{Y} \in \mathbb{R}^{n\times p}$ denote n p-dimensional vectors of feature variables generated by a linear combination of latent variables $\mathbf{Z}$. As a special case, the MCFA model further reduces the MFA parameters by setting up a common component factor loading $\mathbf{A} \in \mathbb{R}^{p\times q}$. Moreover, the common loading can be considered as a transformation that reduces the p-dimensional space to a latent q-dimensional one. For the observed random samples, $\mathbf{y}_1, \ldots, \mathbf{y}_n$ the MCFA model becomes a mixtures of Gaussians with constrained mean and covariance as defined in Eq.(3.9) and is given by

$$P(\mathbf{y};\boldsymbol{\theta}_i) = \pi_i \prod_{j=1}^{n} \mathcal{N}(\mathbf{y}_j; \mathbf{A}\boldsymbol{\xi}_i, \mathbf{A}\boldsymbol{\Omega}_i\mathbf{A}^T + \boldsymbol{\Psi}) \ , \tag{3.21}$$

In the above, $\boldsymbol{\xi}_i$ is a q-dimensional vector, and $\boldsymbol{\Omega}_i$ is a $q \times q$ positive definite matrix, and probability π_i, with $i = 1, \ldots, g$ being the component indicator. The independent noise variance matrix $\boldsymbol{\Psi}$ is a global parameter. where $\boldsymbol{\theta}_i = \{\pi_i, \mathbf{A}, \boldsymbol{\xi}_i, \boldsymbol{\Omega}_i, \boldsymbol{\Psi}\}_{i=1}^{g}$ are the model parameters. Each component can be modelled through a Gaussian distribution $\mathcal{N}(\mathbf{y}_j; \boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2)$. Given the mixture of g components, with ω_{ij} denoting the binary component indicator that is one if and only if the j^{th} object belongs to the i^{th} component, the posterior can be expressed with Bayes theorem as

$$P(\omega_i \mid \mathbf{y}_j;\boldsymbol{\theta}) = \tau_i(\mathbf{y}_j;\boldsymbol{\theta}_i) = \frac{\pi_i\mathcal{N}(\mathbf{y}_j;\boldsymbol{\theta}_i)}{\sum_{h=1}^{g}\pi_h\mathcal{N}(\mathbf{y}_j;\boldsymbol{\theta}_h)} \ . \tag{3.22}$$

Since the latent variables $\mathbf{Z}_{i1}, \ldots, \mathbf{Z}_{in}$, are distributed independently as in Eq.(3.7), the probability density function is $P(\mathbf{Z}_{ij} \mid \omega_i) = \mathcal{N}(\mathbf{Z}_{ij} \mid \boldsymbol{\xi}_i, \boldsymbol{\Omega}_i)$.

For the training stage, the model parameters can be determined via maximum-likelihood using the EM algorithm [40, 61]. The log-likelihood of the model is given by Eq. (3.12). Therefore, the parameters $\boldsymbol{\theta}$ can be optimised by maximising the expected log-likelihood $\mathbb{E}_{\tau_i}[\log \mathcal{L}(\boldsymbol{\theta})]$. The detailed algorithm can be found in 3.1.2.

3.3.3 Main Model

Let us consider the construction of a 2L-MJFA with two hidden layer factors, with these factors sharing a common factor loading. For classification, the observation data are known as $\mathbf{Y} = [\mathbf{Y_1}; \ldots; \mathbf{Y_m}]$, where $\mathbf{Y}_l = [\mathbf{y}_1^l; \ldots; \mathbf{y}_{l_n}^l]$, and $l = 1, \ldots, m$ indicates all the data of the l^{th} class. In our model, the 1^{st} layer defines a normal mixtures of factor analysers with common loading, where each component represents a class, as

$$\mathbf{y}_j^l = \mathbf{A}\mathbf{U}_j^l + \mathbf{e}_j^l, \qquad j = 1, \ldots, l_n, \quad \sum_{l=1}^{m} l_n = n \, . \tag{3.23}$$

In the above, l_n denotes the n^{th} observation belonging to l^{th} class, and $\mathbf{U}_j^l$ denotes the hidden variables. $\mathbf{A} \in \mathbb{R}^{p \times q}$ is a joint factor loading to fit all classes of observations, which can also be considered to be the transformation matrix that projects each pattern to a q-dimensional latent space. $\mathbf{e}_j^l$ denotes the Gaussian noise term for the l^{th} class.

The 2^{nd} layer of 2L-MJFA representing each class consists of an unspecified number of mixtures. The key point here is that the joint factor loading $\mathbf{A}$ is also used as a common loading that is shared across all the components in each class. Then all the observations can be generated by a joint learning model with latent variables $\mathbf{Z}_{ij}^l \sim \mathcal{N}(\boldsymbol{\xi}_i^l, \boldsymbol{\Omega}_i^l)$ of all classes.

For the observation vectors $\mathbf{y}_j^l$ belonging to each class l, the model can then be described as

$$\mathbf{y}_j^l = \mathbf{A}\sum_{i=1}^{g} \mathbf{Z}_{ij}^l + \mathbf{e}_j^l \, , \tag{3.24}$$

where $j = 1, \ldots, l_n$, and $i = 1, \ldots, g$. l_n denotes the n^{th} observation belonging to l^{th} class, and $\mathbf{e}_j^l$ the random noise distributed independently under $\mathcal{N}(0, \boldsymbol{\Psi})$, where $\boldsymbol{\Psi}$ is diagonal. This novel setting implies that each specific class is assumed to be an MCFA model, whereas a joint factor loading exists for all the MCFA models across all data classes. Specifically, the model shares a joint factor loading for all the classes and this is potentially beneficial to both feature extraction and classification, especially in S3 situations.

We now calculate the total number of parameters involved in 2L-MJFA. Since we share a single loading matrix across all the components, the total number of parameters is

$$N_3 = pq - q^2 + p + mg\left[1 + q + \frac{q(q+1)}{2}\right] ,$$

where $pq - q^2$ is the number of parameters in $\mathbf{A}$, and p the parameters of the diagonal matrix $\mathbf{D}$. The mMCFA offers a great reduction in the parameters of the loading $\mathbf{A}$ for each component. Compared with mMCFA, the proposed model significantly reduces the parameter number by $(m-1)(pq - q^2 + p)$.

3.3.4 Optimisation via a Modified EM Algorithm

The proposed 2L-MJFA model is composed of two layers of mixtures of Gaussians. The overall distribution for the mixture of mixtures is the joint distribution of their components given as

$$P(\mathbf{y}_j^l;\boldsymbol{\theta}) = \sum_{l=1}^{m}\pi_l \prod_{j=1}^{l_n} P(\mathbf{y}_j^l;\boldsymbol{\theta}) , \tag{3.25}$$

where $\boldsymbol{\theta} = \{\pi_i, \mathbf{A}, \boldsymbol{\xi}_i^l, \boldsymbol{\Omega}_i^l, \boldsymbol{\Psi}\}$. Actually, the 2^{nd} layer of each class is an MCFA model, which can be easily written as the multivariate Gaussian distribution of Eq.(3.21). For inference, the conditional expectation of the component indicators ω_i^l with $i = 1, \ldots, g$ and $l = 1, \ldots, m$, can be regarded as the posterior probability $P_{\boldsymbol{\theta}}\{\omega_i^l = 1 \mid \mathbf{y}_j^l\}$, implying that $\mathbf{y}_j^l$ belongs to the i^{th} component of class l. With the above definitions, we obtain the conditional distribution $P(\mathbf{y}_j^l \mid \mathbf{U}_{ij}^l) = \mathcal{N}(\mathbf{y}_j^l \mid \mathbf{A}\mathbf{U}_{ij}^l, \boldsymbol{\theta})$. The posterior over all components can then be obtained as

$$\mathbb{E}_\theta\{\omega_i^l \mid \mathbf{y}_j^l\} = Pr_{\boldsymbol{\theta}}\{\omega_i^l = 1 \mid \mathbf{y}_j^l\} = \tau_i^l(\mathbf{y}_j^l;\boldsymbol{\theta}) , \tag{3.26}$$

where

$$\tau_i^l(\mathbf{y}_j^l;\boldsymbol{\theta}) = \frac{\pi_l P(\mathbf{y}_j^l;\boldsymbol{\theta})}{\sum_{h=1}^{m}\pi_h P(\mathbf{y}_j^l;\boldsymbol{\theta})} .$$

Maximum likelihood learning of 2L-MJFA can be conducted with a modified EM algorithm. Within the modified EM framework, the global log-likelihood function of the model is given by

$$\begin{aligned}\log \mathcal{L}_l(\boldsymbol{\theta}) = \sum_{l=1}^{m}\sum_{i=1}^{g}\sum_{j=1}^{n}\omega_i^l\Big\{ &\log \pi_l + \log\phi(\mathbf{y}_j^l;\mathbf{A}\mathbf{U}_{ij}^l,\boldsymbol{\Psi}) \\ &+ \log\phi(\mathbf{U}_{ij}^l;\boldsymbol{\xi}_{ij}^l,\boldsymbol{\Omega}_{ij}^l)\Big\} ,\end{aligned} \tag{3.27}$$

where

$$\phi(\mathbf{y}_j^l;\boldsymbol{\theta}) = \sum_{i=1}^{g} \pi_i^l \mathcal{N}(\mathbf{y}_j^l;\boldsymbol{\theta}) \ .$$

Different from the alternating expectation-conditional maximisation algorithm (AECM) [99], the M-step of the modified EM algorithm is turned into two layer loops. The outer loop is used to update the global parameters $\mathbf{A}$ and $\mathbf{D}$, and the other parameters within each specific class are updated in the inner loop. The training of the above two layers alternate, so that a local optimum could be finally achieved. The overall EM training procedure is summarised in Algorithm 1, and specifics for each stage are explained in the following subsections.

Algorithm 1: EM learning for 2L-MJFA.

```
Input          : A training set with $m$ classes $[\mathbf{Y}_1; \dots; \mathbf{Y}_m]$ and a test set
                 $\mathbf{T} \in \mathbb{R}^{N\times P}$.
Output         : Optimal values of parameters $\boldsymbol{\theta}$.
Initialisation : Set $\theta = \{\pi, \mathbf{A}, \boldsymbol{\xi}, \boldsymbol{\Omega}, \boldsymbol{\Psi}\}$, and evaluate the initial value of the
                 log-likelihood.
Repeat
    E-step :
        Exploit the current parameter values to approximate the posterior
        expectations with Eqs.(3.32,3.33): $\mathbb{E}(\mathbf{Z} \mid \mathbf{y}_j^l, \omega_{ij}^l)$ and $\mathbb{E}(\mathbf{Z}\mathbf{Z}^T \mid \mathbf{y}_j^l, \omega_{ij}^l)$.
    for $l = 1$ to $m$ do
        M-step :
            Update $\mathbf{A}$ and $\boldsymbol{\Psi}$.
            Re-estimate the parameters $\mathbf{A}, \boldsymbol{\Psi}$ using the current responsibilities with
            Eqs.(3.35,3.36), by solving a set of liner equations: $\frac{\partial Q(\boldsymbol{\theta};\boldsymbol{\theta}^{(k)})}{\partial \mathbf{A}} = 0$,
            $\frac{\partial Q(\boldsymbol{\theta};\boldsymbol{\theta}^{(k)})}{\partial \boldsymbol{\Psi}} = 0$.
            Update $\{\pi_i^l, \boldsymbol{\xi}_i^l, \boldsymbol{\Omega}_i^l\}$.
            for $i = 1$ to $g$ do
                Re-estimate the parameters $\pi_i^l, \boldsymbol{\xi}_i^l, \boldsymbol{\Omega}_i^l$ by solving the equations
                $\pi_i^{(k+1)} = \frac{1}{n_l}\sum_{j=1}^{l_n} \tau_{ij}^{(k)}$, $\frac{\partial Q(\boldsymbol{\theta};\boldsymbol{\theta}^{(k)})}{\partial \boldsymbol{\xi}_i} = 0$ and $\frac{\partial Q(\boldsymbol{\theta};\boldsymbol{\theta}^{(k)})}{\partial \boldsymbol{\Omega}_i} = 0$ for each
                class.
Until Convergence
```

Then, we introduce the convergence proof of the modified EM algorithm. By the observations $\mathbf{y}^l$, the latent factor $\mathbf{U}^l$ and the parameter $\boldsymbol{\theta}$, the log-likelihood function of each class can be written as

$$\log \mathcal{L}(\boldsymbol{\theta}) \;=\; \log \int_{\mathbf{U}^l} p(\mathbf{y}^l, \mathbf{U}^l|\boldsymbol{\theta}) d\mathbf{U}^l \ . \tag{3.28}$$

Based on the Jensen inequality, it is easy to derive the lower bound $F(l)$ where $q(\mathbf{U}^l)$ denotes the posterior of the latent variable,

$$\begin{aligned}\log \mathcal{L}(\boldsymbol{\theta}) &= \log \int_{\mathbf{U}^l} q(\mathbf{U}^l) \frac{p(\mathbf{y}^l, \mathbf{U}^l|\boldsymbol{\theta})}{q(\mathbf{U}^l)} d\mathbf{U}^l \\ &\geq \int_{\mathbf{U}^l} q(\mathbf{U}^l) \log \frac{p(\mathbf{y}^l, \mathbf{U}^l|\boldsymbol{\theta})}{q(\mathbf{U}^l)} d\mathbf{U}^l = F(l) \ ,\end{aligned} \tag{3.29}$$

$$F(l) = \int_{\mathbf{U}^l} q(\mathbf{U}^l) \log p(\mathbf{y}^l, \mathbf{U}^l|\boldsymbol{\theta}) d\mathbf{U}^l - \int_{\mathbf{U}^l} q(\mathbf{U}^l) \log q(\mathbf{U}^l|\mathbf{y}^l; \boldsymbol{\theta}) d\mathbf{U}^l \ . \tag{3.30}$$

More concretely, the variational lower bound on the whole model **F** can be written as the sum of class's lower bounds,

$$\mathbf{F} = \sum_{l=1}^{m} F(l) = \sum_{l=1}^{m} E_{q(U^l)}[\log p(\mathbf{y}^l, \mathbf{U}^l|\boldsymbol{\theta})] + H(q) \ . \tag{3.31}$$

Here $H(q)$ denotes the information entropy and $H(q) \geq 0$, thus EM usually maximises the expectation part. In simple terms, the only difference process of the modified EM algorithm is to alternatively construct class's lower bounds and estimate the local parameters by optimising this function when the global parameters fixed. It is easy to prove that any increase in the class's lower bound will lead to an increase in the true likelihood of the model. Finally, with a multi-linear Gaussian likelihood model, the per-iteration complexity of the proposed model in E-step is calculated as $O(n(p + mgq + 1))$ and that in M-step is $O(n(m(p + 1) + g(q + 1)))$.

E-step

In this step, Eq. (3.22) is used to compute the posterior over the latent variables. Given the current setting of the model parameters, the expectations of the hidden variables $\mathbb{E}(\mathbf{Z} \mid \mathbf{y}_j^l, \omega_{ij}^l)$ and $\mathbb{E}(\mathbf{Z}\mathbf{Z}^T \mid \mathbf{y}_j^l, \omega_{ij}^l)$ are easily verified as the following derivations for all the data points $j = 1, \ldots, l_n$ and mixture components $i = 1, \ldots, g$ can be produced as

$$\mathbb{E}(\mathbf{Z} \mid \mathbf{y}_j^l, \omega_{ij}^l) = \boldsymbol{\xi}_i^l + \boldsymbol{\gamma}_i^T \mathbf{y}_{ij} \ , \tag{3.32}$$

and

$$\mathbb{E}(\mathbf{Z}\mathbf{Z}^T \mid \mathbf{y}_j^l, \omega_{ij}^l) = (\mathbf{I}_q - \boldsymbol{\gamma}_i^T \mathbf{A}) \boldsymbol{\Omega}_{ij}^l + \mathbb{E}(\mathbf{Z} \mid \mathbf{y}_j^l, \omega_{ij}^l) \mathbb{E}(\mathbf{Z} \mid \mathbf{y}_j^l, \omega_{ij}^l)^T \ , \tag{3.33}$$

where

$$\begin{aligned}\mathbf{y}_{ij} &= \mathbf{y}_j^l - \mathbf{A}\boldsymbol{\xi}_i^l \ , \\ \boldsymbol{\gamma}_i &= (\mathbf{A}\boldsymbol{\Omega}_i^l \mathbf{A}^T + \boldsymbol{\Psi})^{-1} \mathbf{A} \boldsymbol{\Omega}_i \ .\end{aligned}$$

For the iteration of each class, $\mathbf{Q}(\boldsymbol{\theta};\boldsymbol{\theta}^{(k)})$ denotes the conditional expectation of log-likelihood as Eq. (3.15), given the observed data $\mathbf{y}$ and $\boldsymbol{\theta}^{(k)}$, where $\boldsymbol{\theta}^{(k)}$ denotes the current estimates. Denoting the posterior $\tau_{ij}^{(k)} = \tau_i^l(\mathbf{y}_j^l;\boldsymbol{\theta}^{(k)})$, we can transform the expectation function as

$$\begin{aligned}\mathbf{Q}(\boldsymbol{\theta};\boldsymbol{\theta}^{(k)}) = \sum_{i=1}^{g}\sum_{j=1}^{l_n}\tau_{ij}^{(k)}\Big\{ \log\pi_i^l + \mathbb{E}_{\boldsymbol{\theta}^{(k)}}\Big[\log\mathcal{N}(\mathbf{y}_j^l;\mathbf{A}\mathbf{Z}_{ij}^l,\boldsymbol{\Psi})|\mathbf{y}_j^l,\omega_{ij}^l=1\Big] \\ + \mathbb{E}_{\boldsymbol{\theta}^{(k)}}\Big[\log\mathcal{N}(\mathbf{Z}_{ij}^l;\boldsymbol{\xi}_i^l,\boldsymbol{\Omega}_i^l)|\mathbf{y}_j,\omega_{ij}^l=1\Big]\Big\}\ .\end{aligned} \tag{3.34}$$

Algorithm 2: Classification procedure for 2L-MJFA.

```
Input: A training set with $m$ classes $[\mathbf{Y}_1;\dots;\mathbf{Y}_m]$ and a test set $\mathbf{T}\in\mathbb{R}^{N\times P}$.
Training phase :
    Initialise the global parameters $\mathbf{A}, \boldsymbol{\Psi}$ based on all the training data.
    Divide each $\mathbf{Y}_l$, for $l = 1,\dots,m$ into $g$ components randomly and then
    initialise the local parameters $\pi_i, \boldsymbol{\xi}_i, \boldsymbol{\Omega}_i$.
    Repeat
        for $l = 1$ to $m$ do
            Estimate the probability of data generated by each component in
            Eq.(3.25) and the posterior probability $P_{\boldsymbol{\theta}}\{\omega_{ij}^l = 1|\mathbf{T}_j\}$, for
            $j = 1,\dots,N$ that $\mathbf{T}_j$ belongs to the $i^{th}$ component by each class in
            Eq. (3.26).
            for $i = 1$ to $g$ do
                Use the alternate EM algorithm, and update local parameters by
                calculating the expectation of log-likelihood in Eq. (3.34) of each
                class.
            Compute the log-likelihood value $\mathcal{L}_l(\boldsymbol{\theta})$ using Eq. (3.27).
    Until $\mathcal{L}_l(\boldsymbol{\theta})^{(new)} - \mathcal{L}_l(\boldsymbol{\theta}) <$ threshold value
Testing phase :
    Compute the posterior probabilities $\tau_l(\mathbf{T}_j;\boldsymbol{\theta})$ of each class with test data.
    Assign each test data point $\mathbf{T}_j$ to the $l$ class for which $\tau_l(\mathbf{T}_j;\boldsymbol{\theta}) \geq \tau_h(\mathbf{T}_j;\boldsymbol{\theta})$ for
    $h = 1,\dots,m$ with $h \neq l$.
```

M-step

In the subsequent step, the updated estimates of the global parameters can be obtained by taking the partial derivatives of expectation log-likelihood function for each parameter. The joint factor loading is updated as

$$\mathbf{A}^{(k+1)} = \left(\sum_{l=1}^{m}\sum_{i=1}^{g}\mathbf{A}_{li(1)}^{(k)}\right)\left(\sum_{l=1}^{m}\sum_{i=1}^{g}\mathbf{A}_{li(2)}^{(k)}\right)^{-1}\ , \tag{3.35}$$

where

$$\mathbf{A}_{li(1)}^{(k)} = \sum_{j=1}^{l_n} \tau_{ij}^{(k)} \left\{ \mathbf{y}_j^l \mathbb{E}^{(k)}(\mathbf{Z} \mid \mathbf{y}_j^l, \omega_{ij}^{l(k)}) \right\} ,$$

$$\mathbf{A}_{li(2)}^{(k)} = \sum_{j=1}^{l_n} \tau_{ij}^{(k)} \left\{ \mathbb{E}^{(k)}(\mathbf{Z}\mathbf{Z}' \mid \mathbf{y}_j^l, \omega_{ij}^{l(k)}) \right\} .$$

The updated estimates of the common diagonal covariance matrix can then be written as

$$\mathbf{\Psi}^{(k+1)} = \frac{1}{n} diag \Big[\sum_{l=1}^{m} \sum_{j=1}^{l_n} \tau_{ij}^{(k)} (\mathbf{\Psi}_1^{(k)} + \mathbf{\Psi}_2^{(k)}) \Big] , \tag{3.36}$$

where

$$\mathbf{\Psi}_1^{(k)} = \mathbf{\Psi}^{(k)}(\mathbf{I}_p - \boldsymbol{\beta}^{(k)}) ,$$
$$\mathbf{\Psi}_2^{(k)} = \boldsymbol{\beta}^{(k)^T} (\mathbf{y}_{ij}^{(k)})(\mathbf{y}_{ij}^{(k)})^T \boldsymbol{\beta}^{(k)} ,$$
$$\boldsymbol{\beta}^{(k)} = \left(\mathbf{A}^{(k)} \mathbf{\Omega}^{(k)} \mathbf{A}^{(k)^T} + \mathbf{\Psi}^{(k)} \right)^{-1} \mathbf{\Psi}^{(k)} ,$$
$$\mathbf{y}_{ij}^{(k)} = \mathbf{y}_j - \mathbf{A}^{(k)} \boldsymbol{\xi}_i^{(k)} .$$

For each class l, the updated estimates $\pi_i^{(k+1)}$, $\boldsymbol{\xi}_i^{(k+1)}$ and $\mathbf{\Omega}_i^{(k+1)}$ can be obtained by calculating the equations $\frac{\partial Q(\boldsymbol{\theta};\boldsymbol{\theta}^{(k)})}{\partial \boldsymbol{\xi}_i} = 0$, $\frac{\partial Q(\boldsymbol{\theta};\boldsymbol{\theta}^{(k)})}{\partial \mathbf{\Omega}_i} = 0$. Specifically, it is easy to verify that $\pi_i^{(k+1)} = \frac{1}{n_l} \sum_{j=1}^{l_n} \tau_{ij}^{(k)}$, for $i = 1, \ldots g$, where n_l denotes the number of observations in l_{th}class. The local parameter updates can be obtained via the following

$$\boldsymbol{\xi}_i^{(k+1)} = \boldsymbol{\xi}_i^{(k)} + \frac{\sum_{j=1}^{l_n} \tau_{ij}^{(k)} \boldsymbol{\varphi}^{(k)}}{\sum_{j=1}^{l_n} \tau_{ij}^{(k)}} , \tag{3.37}$$

$$\mathbf{\Omega}_i^{(k+1)} = \frac{\sum_{j=1}^{l_n} \tau_{ij}^{(k)} \boldsymbol{\varphi}^{(k)} \boldsymbol{\varphi}^{(k)^T}}{\sum_{j=1}^{l_n} \tau_{ij}^{(k)}} + (\mathbf{I}_q - \boldsymbol{\varphi}^{(k)}) \mathbf{\Omega}_i^{(k)} , \tag{3.38}$$

$$\boldsymbol{\varphi}^{(k)} = \boldsymbol{\gamma}_i^{(k)^T} \mathbf{y}_{ij}^{(k)} .$$

Algorithm 2 summarises the overall classification procedure.

3.3.5 Experiments and Results

To demonstrate the effectiveness of our proposed algorithm, we conduct extensive experiments on a variety of datasets. We compare our two-layer mixture approach with three other competitive methods. Specifically, we compare it with mMCFA, mPPCA, and the independent learning approaches of PCA followed by GMM (PCA-GMM), and LDA followed by GMM (LDA-GMM).[4] Unlike hard assignment methods (e.g. k-means), GMM

[4]PCA or LDA are first used to perform DR and then a GMM is used for the classification.

is a soft assignment method which gives the probability that the data points are assigned to each class, rather than just giving a definitive class membership [118]. Obtaining a probability is beneficial as it provides confidence for the results. The used datasets include a synthetic one, an ordinary one, and five S3 datasets. We report the error rate (ERR) of the classification in terms of different reduced dimensionalities for the various algorithms on the test data. All the experimented methods are implemented in the MATLAB platform.

Synthetic dataset To illustrate the advantage of the joint learning in the proposed model, we generate synthetic data to visualise the obtained subspaces for PCA, MCFA and the 2L-MJFA. The synthetic dataset consists of 2 classes of 32-dimensional samples. For each class, the first two dimensions are randomly generated by a multivariate normal distribution with means and covariance set to

$$\mu_1 = (3.2875, 3.4905)^T, \qquad \mu_2 = (2.9185, 2.9732)^T,$$

$$\Sigma_1 = \begin{pmatrix} 23.2368 & 19.2956 \\ 19.2956 & 19.8985 \end{pmatrix}, \quad \Sigma_2 = \begin{pmatrix} 5.0030 & 0.8919 \\ 0.8919 & 4.4236 \end{pmatrix}.$$

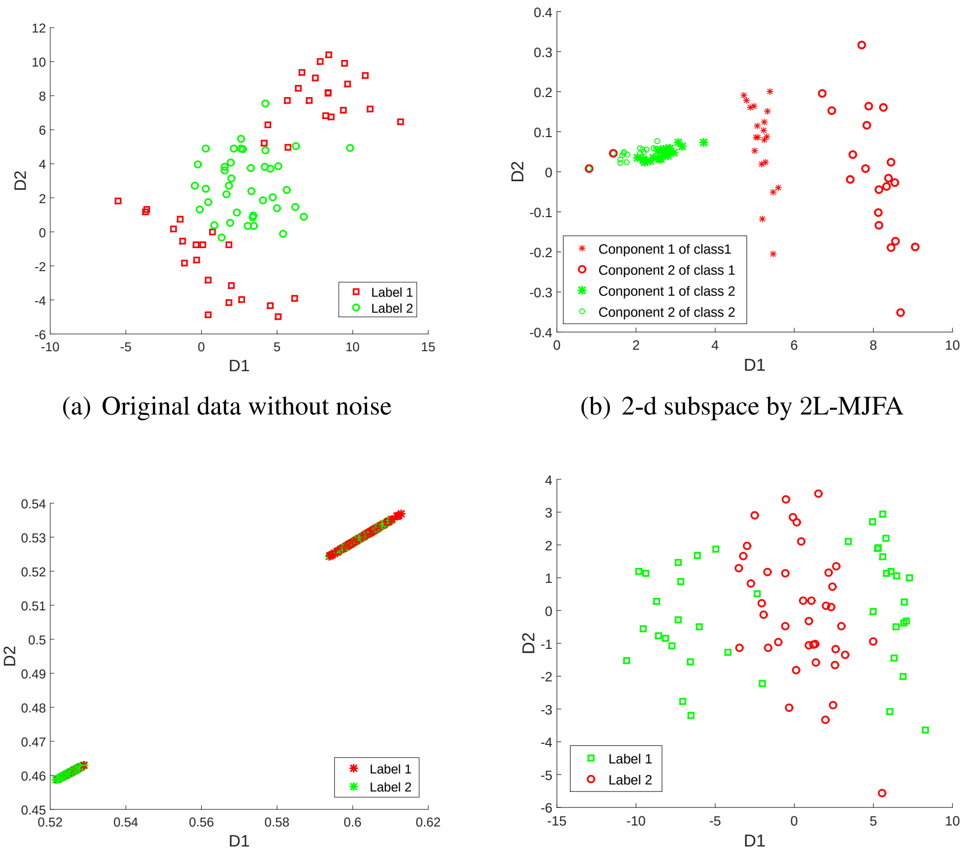
The other 30-dimensions are generated as random Gaussian noise.

The obtained 2-dimensional subspaces are visualised in Fig. 3.13. The top-left of the figure shows the ground truth samples without the additional 30-dimensional noisy features. It can be clearly seen that the class denoted by label 1 consists of two modalities. The proposed 2L-MJFA shows to perform better than the other two, as its subspace demonstrates a much better separability than PCA and MCFA. The mPPCA does not map all the data in a subspace, since the approach is used to classification by building an mPPCA model of each class, which means that the patterns for different classes are mapped into different subspaces. Also, LDA can generate subspaces up to $m - 1$ dimensions, which is one dimension for the current dataset.

User knowledge data

The employed User Knowledge dataset describes students' knowledge status about the subject of Electrical DC Machines [119]. This dataset consists of 403 training samples and 206 test samples. Each sample is of 40 dimensions with 5 being attribute information, plus 35 random noisy features. The class labels correspond to four student knowledge levels. We compare the 2L-MJFA and other mixture joint learning methods against different reduced dimensionalities ranging from 1 to 20.

We report the comparative results in Table 3.8. We can see that the mixture joint learning methods 2L-MJFA and mMCFA provide the lowest error rates. In particular,



(a) Original data without noise (b) 2-d subspace by 2L-MJFA

(c) 2-d subspace by MCFA (d) 2-d subspace by PCA

Fig. 3.13: Visualisation of DR for 2L-MJFA, MCFA, and PCA on simulated data, where (1) is the ground truth. Different patterns represent different classes, and different shapes within the same grey scale indicate different class modalities.

Table 3.8: Error rate comparison for various dimensions, on the User Knowledge dataset.

Dimension:	1	3	5	10	15	20
2L-MJFA	0.1214	0.0689	**0.0414**	0.0620	0.0620	0.0620
mMCFA	0.2276	0.0552	0.0896	0.0758	0.1517	0.2827
mPPCA	0.3172	0.2897	0.2690	0.1931	0.1586	0.0897
PCA-GMM	0.6621	0.2483	0.2345	0.1214	0.1931	0.2966
LDA-GMM	0.4000	0.3724	-	-	-	-

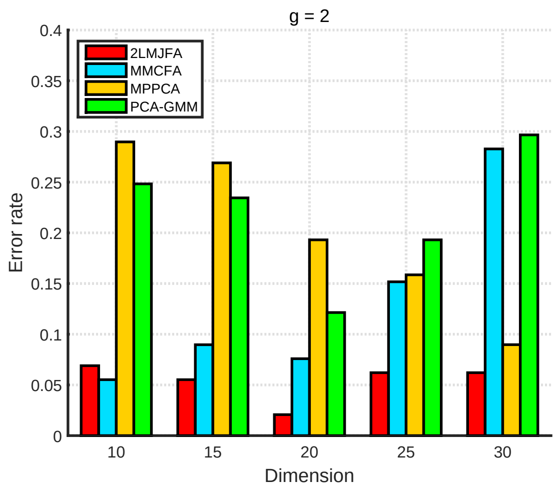


Fig. 3.14: Error rate comparison for the User Knowledge dataset.

when the dimensionality is reduced to 5 (the actual dimension), 2L-MJFA yields the best performance with the error rate being 0.0414. This is significantly lower than mMCFA, mPPCA and PCA-GMM. LDA-GMM just allows reducing dimensionality to $1-3$, since this dataset has $m = 4$ classes. From the results, LDA does not provide an optimised subspace for test data. To better illustrate the performance, we also plot the results in Fig.3.14, where it can be seen that 2L-MJFA outperforms the other algorithms in most cases.

Small sample size datasets

In this subsection, we compare the proposed 2L-MJFA with the various other algorithms across five S3 datasets.

Experimental Setup We evaluate the performance of the various algorithms by using a 5-fold cross-validation on the five S3 datasets, which are WDBC, WPBC, ULC, LSVT and BT. To make the problems more challenging, we intentionally use one of the five partitions as the training set, while the remaining four partitions as the testing set. The average error rate on the test sets is then reported for varying mixture numbers and reduced dimensionality. Table 3.9 summarises the statistics of these five S3 datasets. As seen in the table, the number of dimensions are sometimes larger than the number of training samples (e.g., in ULC and LSVT).

Breast cancer Wisconsin dataset This dataset contains two subsets, the Wisconsin diagnostic breast cancer (WDBC) and the Wisconsin prognostic breast cancer (WPBC) [120, 121]. WDBC contains 569 instances which are divided into the two diagnostic

Table 3.9: Summary of S3 datasets.

Dataset:	Training samples:	Test samples:	Dimensions:	Classes:
WDBC	114	455	60	2
WPBC	38	156	33	2
ULC	77	273	148	3
LSVT	56	42	309	2
BT	81	24	39	6

predictions of benign and malignant. The 60 attributes consist of 30 real-valued input features and 30 additional Gaussian noise features. WPBC contains 194 instances, which record two classes of patients, that is being recurrent or not post-surgical.

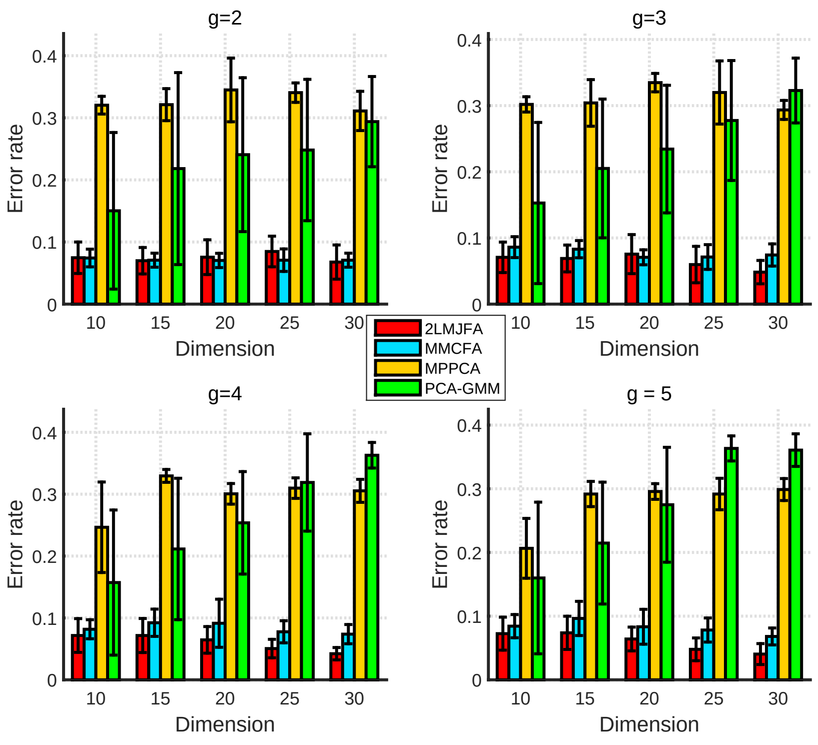


Fig. 3.15: Error rate comparison for the WDBC dataset.

Results on the Wisconsin diagnostic breast cancer (WDBC) Table 3.10 shows the error rate comparison from reducing the dimensions from 10 to 30 and setting each class to $g = 2$ to 5 mixture components for different subspaces (DIM). For LDA-GMM, the dimensionality is just allowed to reduce to 1, because there are 2 classes in these two datasets. We can find that the error rate of 2L-MJFA decreases as the number of mixture

Table 3.10: Error rate comparison for the WDBC dataset.

WDBC						
DIM	g	2L-MJFA	mMCFA	mPPCA	PCA-GMM	LDA-GMM
1	2	0.1023±0.02	0.0703±0.02	0.2846±0.03	0.0935 ±0.03	0.0350±0.01
	3	0.1010±0.01	0.0686±0.02	0.2509±0.03	0.0935±0.03	$\mathbf{0.0282 \pm 0.01}$
	4	0.1022±0.01	0.0703±0.03	0.2778±0.04	0.0935±0.03	0.0334±0.01
	5	0.1076±0.01	0.0705±0.02	0.2759±0.01	0.0935±0.03	0.0334±0.01
10	2	0.0746±0.02	0.0742±0.01	0.3202±0.01	0.1502±0.12	-
	3	0.0707±0.02	0.0861±0.02	0.3019±0.02	0.1528±0.15	-
	4	0.0716±0.02	0.0817±0.02	0.2465±0.05	0.1571±0.12	-
	5	0.0441±0.03	0.0842±0.02	0.2065±0.03	0.1600±0.11	-
15	2	0.0698±0.02	0.0707±0.01	0.3212±0.03	0.2182±0.15	-
	3	0.0689±0.02	0.0830±0.01	0.3041±0.03	0.2050±0.10	-
	4	0.0716±0.03	0.0922±0.02	0.3295±0.04	0.2114±0.11	-
	5	0.0737±0.03	0.0963±0.03	0.2917±0.03	0.2147±0.09	-
20	2	0.0755±0.02	0.0703±0.02	0.3448±0.05	0.2406±0.12	-
	3	0.0755±0.03	0.0707±0.01	0.3348±0.06	0.2343±0.08	-
	4	0.0645±0.02	0.0914±0.04	0.3005±0.06	0.2536±0.08	-
	5	0.0641±0.01	0.0833±0.03	0.2956±0.02	0.2749±0.09	-
25	2	0.0680±0.02	0.0707±0.01	0.3405±0.02	0.2481±0.11	-
	3	0.0597±0.03	0.0712±0.02	0.3199±0.07	0.2775±0.06	-
	4	0.0505±0.01	0.0776±0.02	0.3097±0.04	0.3189±0.07	-
	5	0.0479±0.04	0.0782±0.02	0.2917±0.02	0.3633±0.02	-
30	2	0.0417±0.02	0.0707±0.01	0.3110±0.03	0.2938±0.07	-
	3	0.0483±0.01	0.0743±0.02	0.2935±0.02	0.3229±0.09	-
	4	0.0422±0.01	0.0738±0.02	0.3053±0.02	0.3628±0.02	-
	5	0.0404±0.04	0.0681±0.01	0.2987±0.02	0.3606±0.03	-

components increases. For clarity, we also plot the results in Fig.3.15, where it can be observed that 2L-MJFA achieves the significantly lowest error rate 0.0404 when the dimension is reduced to 30 and the number of components is set to 5. The best result of the competitors is just 0.0279 given by LDA-GMM.

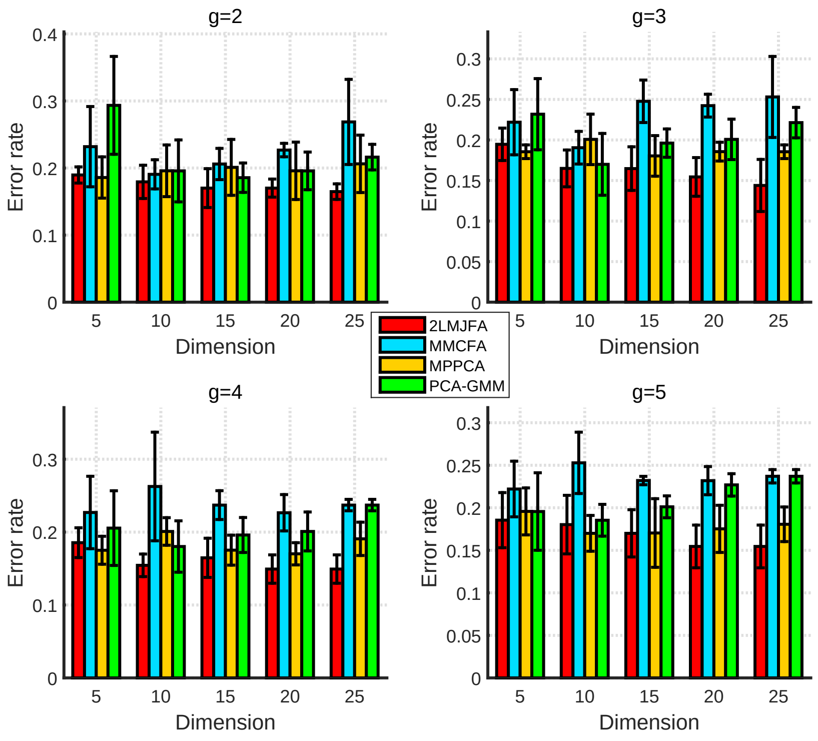


Fig. 3.16: Error rate comparison for the WPBC dataset.

Results on the Wisconsin prognostic breast cancer (WPBC) The results of this comparison are shown in Table 3.11 and Fig. 3.16. We can clearly observe that the 2L-MJFA again achieves the overall best performance. In particular, the 2L-MJFA achieves the lowest error rate 0.1493 when the dimension is reduced to 25; this is significantly lower than the error of 0.1702 from MCFA.

Results on the Urban land cover dataset (ULC) The ULC dataset contains nine types of urban land cover from high-resolution aerial imagery [106, 107]. In this experiment, for simplicity, we only extract three types of experimental data, that is building, concrete, and grass. The number of components g are assumed to be between 2 and 5.

Table 3.12 reports the results across different dimensionalities ranging from 10 to 30 (1 to 2 for LDA-GMM). The best result of 0.1099 is achieved by 2L-MJFA model, for 15 dimensions and 5 components. The other methods perform worse, especially as

Table 3.11: Error rate comparison for the WPBC dataset.

WPBC						
DIM	g	2L-MJFA	mMCFA	mPPCA	PCA-GMM	LDA-GMM
1	2	0.2498±0.03	0.2943±0.10	0.3351±0.07	0.25644±0.03	0.2479±0.05
	3	0.2621±0.03	0.3045±0.14	0.2869±0.11	0.25644±0.03	0.2166±0.02
	4	0.2459±0.02	0.3947±0.12	0.2631±0.03	0.25644±0.03	0.2166±0.04
	5	0.2604±0.03	0.2887±0.13	0.2730±0.03	0.25644±0.03	0.1860±0.03
5	2	0.1896±0.01	0.2319±0.06	0.1859±0.03	0.2935±0.07	-
	3	0.1946±0.02	0.2219±0.04	0.1855±0.01	0.2318±0.04	-
	4	0.1854±0.02	0.2269±0.04	0.1751±0.02	0.2055±0.05	-
	5	0.1854±0.03	0.2220±0.03	0.2250±0.03	0.1956±0.05	-
10	2	0.1793±0.02	0.1906±0.02	0.1929±0.04	0.1957±0.05	-
	3	0.1649±0.02	0.1904±0.02	0.2007±0.03	0.1700±0.02	-
	4	0.1544±0.01	0.2625±0.07	0.2009±0.02	0.1802±0.04	-
	5	0.1802±0.02	0.2528±0.04	0.2000±0.02	0.1853±0.02	-
15	2	0.1700±0.03	0.2060±0.02	0.2010±0.04	0.1856±0.02	-
	3	0.1647±0.02	0.2477±0.02	0.1804±0.03	0.1961±0.02	-
	4	0.1647±0.02	0.2370±0.02	0.1752±0.02	0.1960±0.02	-
	5	0.1699±0.02	0.2320±0.00	0.1750±0.04	0.2011±0.01	-
20	2	0.1700±0.01	0.2268±0.01	0.1959±0.04	0.1957±0.03	-
	3	0.1544±0.02	0.2423±0.01	0.1856±0.01	0.2007±0.02	-
	4	0.1493±0.02	0.2265±0.02	0.1702±0.02	0.2009±0.03	-
	5	0.1545±0.02	0.2319±0.02	0.2000±0.03	0.2267±0.01	-
25	2	0.1648±0.01	0.2687±0.06	0.2063±0.01	0.2163±0.02	-
	3	$\mathbf{0.1493 \pm 0.02}$	0.2531±0.04	0.1855±0.01	0.2214±0.06	-
	4	$\mathbf{0.1493 \pm 0.02}$	0.2370±0.01	0.1907±0.02	0.2370±0.01	-
	5	0.1545±0.02	0.2370±0.01	0.1806±0.02	0.2370±0.01	-

Table 3.12: Error rate comparison for the ULC dataset.

ULC						
DIM	g	2L-MJFA	mMCFA	mPPCA	PCA-GMM	LDA-GMM
2	2	0.5108	0.1209	0.5128	0.3301	0.6557
10	2	0.1319	0.1355	0.4945	0.2491	-
	3	0.1282	0.1282	0.1832	0.2015	-
	4	0.1355	0.1502	0.3736	0.2564	-
	5	0.1245	0.1319	0.2418	0.2418	-
15	2	0.1172	0.3077	0.3846	0.3700	-
	3	0.1172	0.2234	0.3773	0.3846	-
	4	0.1392	0.2381	0.2418	0.3773	-
	5	**0.1099**	0.1722	0.3223	0.4139	-
20	2	0.1209	0.4725	0.3773	0.3883	-
	3	0.1209	0.3919	0.3443	0.4139	-
	4	0.1245	0.3883	0.3883	0.3956	-
	5	0.1172	0.3004	0.3516	0.4396	-
25	2	0.1172	0.4579	0.3114	0.4066	-
	3	0.1392	0.3150	0.2454	0.4176	-
	4	0.1319	0.3810	0.3480	0.4066	-
	5	0.1319	0.4029	0.2930	0.4432	-
30	2	0.1245	0.4286	0.4249	0.4432	-
	3	0.1209	0.2454	0.3443	0.4396	-
	4	0.1429	0.3883	0.2527	0.4505	-
	5	0.1392	0.3883	0.3077	0.4945	-

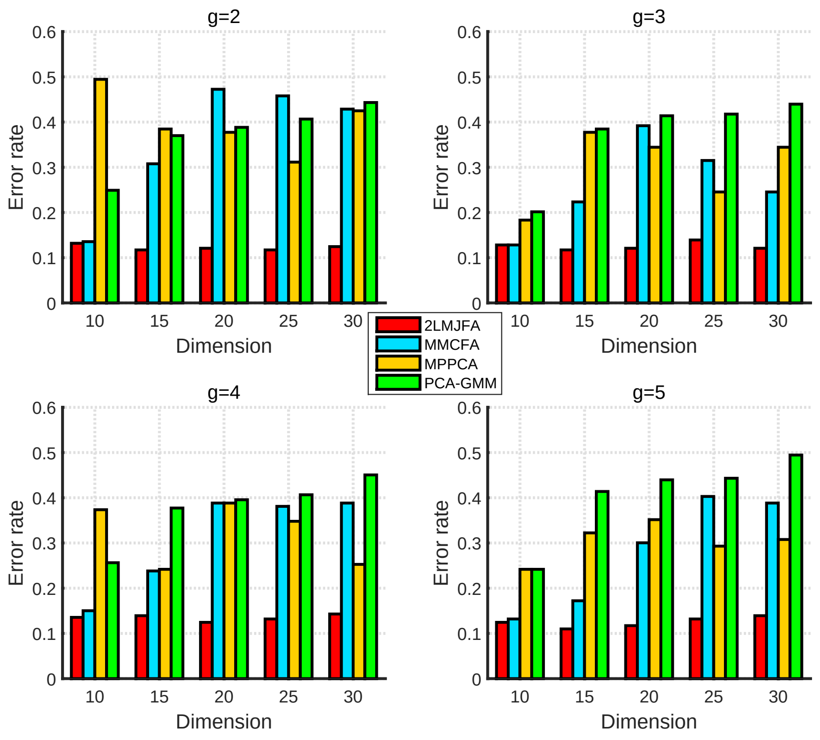


Fig. 3.17: Error rate comparison for the ULC dataset.

the numbers of components and dimensions increase. mMCFA achieves better than the remaining methods. The errors are also summarised in Fig. 3.17.

Results on the LSVT voice rehabilitation dataset (LSVT) The LSVT contains 98 instances with 309 attributes and is used for evaluating whether a phonation considered acceptable or not after voice rehabilitation [122]. The results of Table 3.13 are reported for different dimensions between 5 and 20 (1 for LDA-GMM). It can be seen, that mMCFA and mPPCA achieve their best performance when the dimensionality is reduced to 10. When the dimensions increase, the performance of different algorithms deteriorates quickly due to a more pronounced $S3$ problem. The proposed 2L-MJFA model again achieves the lowest error rate of 0.1792 (when the dimension is set to 20). Fig. 3.18 summarises these errors.

Results on the Breast tissue dataset (BT) This dataset [123] contains 106 objects described by 9 features. For each object, a group of features are selected from excised breast tissue samples using electrical impedance measurement. Six major diagnostic classes are involved that consist of 4 normal breast tissues: connective, glandular, Fibro-adenoma and adipose tissue, as well as 2 pathological tissues, that is: mastopathy and carcinoma.

Table 3.13: Error rate comparison for the LSVT dataset.

LSVT					
DIM	2L-MJFA	mMCFA	mPPCA	PCA-GMM	LDA-GMM
1	0.3171±0.07	0.2897±0.12	0.3731±0.07	0.4019±0.06	0.4246±0.04
5	0.2143±0.08	0.2103±0.10	0.2143±0.04	0.2659±0.07	-
10	0.2023±0.06	0.1980±0.07	0.1964±0.06	0.2698±0.03	-
15	0.1984±0.06	0.2421±0.06	0.2183±0.06	0.2857±0.05	-
20	**0.1792 ± 0.06**	0.2659±0.05	0.2857±0.03	0.2857±0.06	-

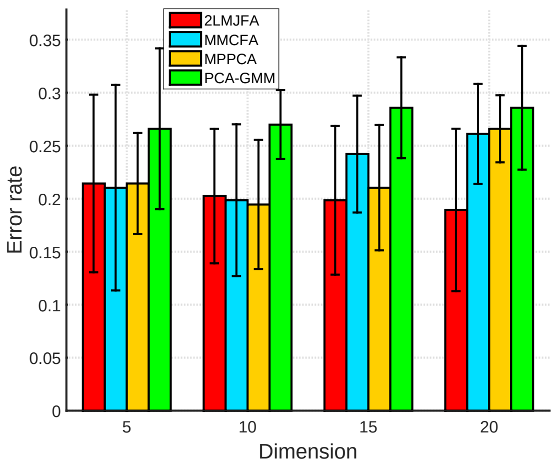


Fig. 3.18: Error rate comparison for the LSVT dataset.

We augment the features to 39 dimensions with random Gaussian noise, in order to accentuate the $S3$ effect. We report the results across different dimensionalities ranging from 2 to 9 (2 to 5 for LDA-GMM) and different component number between 2 and 5. It is worth noting that there are at most 21 samples for each class, which is less than the 39 dimensions.

Table 3.14: Error rate comparison for the BT dataset.

BT						
g	DIM	2L-MJFA	mMCFA	mPPCA	PCA-GMM	LDA-GMM
2	2	0.2468±0.05	0.3902±0.07	0.6692±0.03	0.6517±0.14	0.5576±0.11
	4	$\mathbf{0.1897 \pm 0.01}$	0.4257±0.05	0.7261±0.03	0.6255±0.12	0.5350±0.09
	6	0.1970±0.02	0.4533±0.05	0.6510±0.09	0.6159±0.13	-
	9	0.2073±0.04	0.4902±0.06	0.6418±0.02	0.5899±0.09	-
3	2	0.2540±0.02	0.3900±0.06	0.6892±0.02	0.6032±0.15	0.5479±0.08
	4	0.2359±0.02	0.4164±0.02	0.6713±0.03	0.6076±0.10	0.5053±0.09
	6	0.2371±0.01	0.4615±0.04	0.6442±0.06	0.6088±0.08	-
	9	0.2085±0.04	0.5457±0.04	0.6088±0.07	0.6573±0.06	-
4	2	0.2530±0.05	0.3616±0.07	0.6986±0.05	0.6043±0.15	0.5279±0.09
	4	0.2528±0.05	0.4164±0.02	0.6345±0.04	0.6182±0.09	0.5550±0.06
	6	0.2560±0.03	0.4995±0.02	0.6219±0.06	0.6585±0.06	-
	9	0.2254±0.02	0.5553±0.05	0.6618±0.04	0.6964±0.03	-
5	2	0.2528±0.03	0.3892±0.06	0.6870±0.03	0.6149±0.12	0.5252±0.08
	4	0.2454±0.03	0.4459±0.04	0.6310±0.03	0.5887±0.09	0.4961±0.08
	6	0.2454±0.01	0.5362±0.04	0.6406±0.02	0.6973±0.07	-
	9	0.2169±0.04	0.5553±0.05	0.6406±0.02	0.6677±0.05	-

Table 3.14 reports the results, where the proposed method outperforms the others. The performance difference is more prominent as the number of components and dimensions increases. Fig.3.19 summarises some errors.

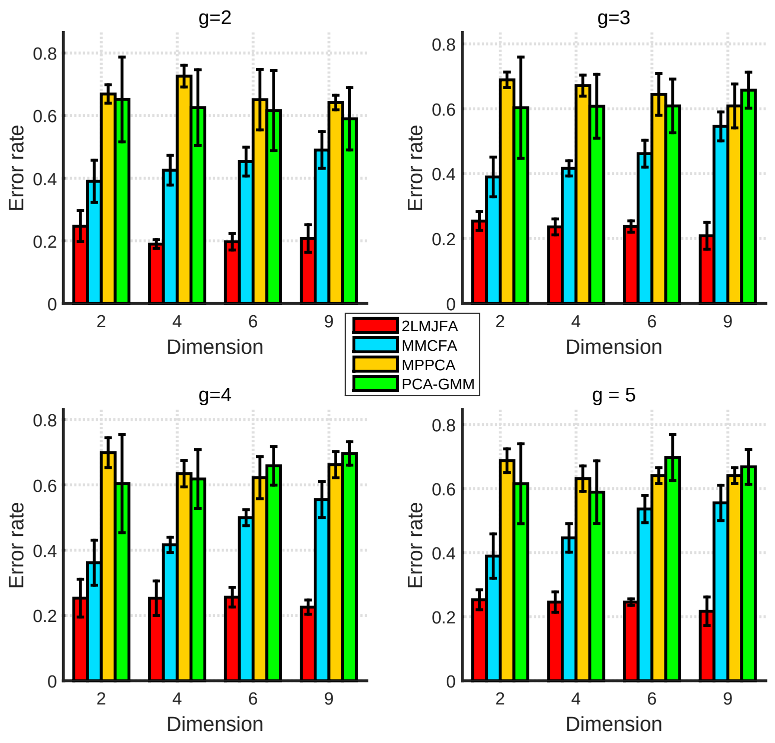


Fig. 3.19: Error rate comparison for the BT dataset.

3.4 Summary

In this chapter, we have introduced an unsupervised method that learning both the optimal subspace and the parameters for GMM jointly and further extended it to a novel two-layer joint learning model, referred to as 2L-MJFA, for supervised learning. These joint learning models are very different from previous approaches, where dimensionality reduction is usually independent of the subsequent parameter learning procedure. Specifically, these models further reduce the parameters of the component-covariance matrices by setting common loading matrix. Moreover, 2L-MJFA is based on a two-layer mixture or a mixture of mixtures structure, with each component that represents each specific class serving as another mixture model of factor analysers designed to share the same loading matrix. The latter has a dual role with respect to being considered a dimensionality reduction matrix and being capable of reducing the model parameters, making, therefore, the proposed algorithm very suitable for S3 problems. We have also described a modified EM algorithm to train the proposed model. For unsupervised learning experiments, the joint learning models have evaluated on both synthetic, real-world datasets and benchmark datasets. The performance of joint learning models is demonstrated to significantly outperforms the independent learning models. Furthermore, the model with a common loading is shown to be the best when compared with other joint models. For supervised learning experiments, a series of experiments have demonstrated that 2L-MJFA significantly outperforms four competitive algorithms. On six real-world datasets, our proposed algorithm has an average decrease of 17.41% compared to the best result of the competitive algorithms.

3.4.1 Discussion

There are several shortcomings with the proposed models in this chapter. On the one hand, both the models need to pre-specify certain parameters beforehand. These parameters include the number of clusters and the reduced dimensionality. In practice, a trial and error procedure can usually be used for searching the best parameters; such procedure is however often computationally slow and is also less elegant. On the other hand, by assuming a common loading, these models exploit a shallow mixture model which may be in lack of expressive ability. This may present difficulties when modelling complicated data.

To mitigate the above mentioned shortcomings, an infinite model could be developed which can determine the number of components and the dimensionality automatically. This exactly motivates the model as discussed in Chapter 4. In addition, aiming to increase the model's expressive capability, we intend to study how to replace the shallow structure with a deep structure which could then lead to an arbitrarily non-linear projection instead of the multi-linear projection as discussed in this chapter. This idea directly motivates the models which will be discussed in Chapter 6.

Chapter 4

Infinite Non-negative Binary Matrix Tri-factorisation for Learning Latent Features

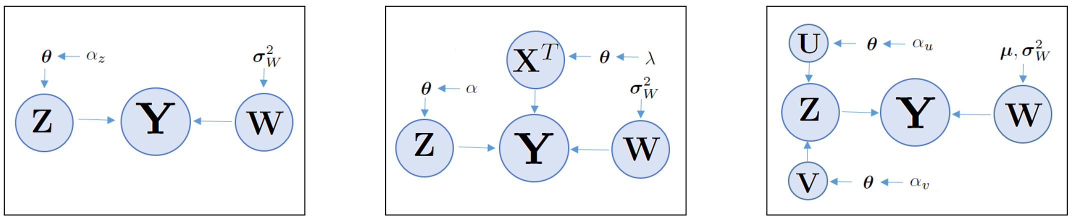
Non-negative matrix factorisation (NMF) is a well-known matrix decomposition technique that has been widely applied in data analysis, and machine learning [124]. Essentially, NMF decomposes the observation data into two matrices, which contain information on the latent features and structures of the object. In other words, NMF can be exploited to reveal latent structures through decomposing the observed variables into two latent variables [125]. The NMF algorithms can be used for feature extraction or one-sided clustering. For instance, Ding et al. [126] proposed a relaxed K-means clustering approach based on NMF. In many real-world applications, people are interested in addressing objects of multiple types with features of much richer structures, also termed dyadic data analysis [8]. To cope with such data, it becomes important to learn the interaction among features and leverages the interrelation between data and features. This is particularly the case in co-clustering or collaborative filtering [127–129]. In such cases, the NMF with two factors can be restrictive and often provides poor low-rank matrix approximation. Thus, a new factor is needed to absorb the additional scale.

To this end, an emerging technique based on Non-negative Matrix Tri-factorisation (NMTF) has recently gained much attention [11]. For instance, Zhang et al. [130] introduced NMTF for biclustering structures, whilst Wang et al. [131] developed a Fast NMTF (FNMTF) method for large-scale data co-clustering. More recently, Wang et al. [132] proposed a Penalty NMTF (PNMT) based approach by introducing three penalty constraints. However, in all these feature-based models, a fixed number of latent features or clusters are generally assumed. To achieve the optimal result, it usually requires this number to be tuned or searched by trial and error. Further, in practice, factor matrices are often required in binary form, since binary features are cheaper to compute and more compact to store. Binary features can also appear in various types of data, such as binary images, and numbers of words occurring in an article [133, 134]. In this scenario, effective NM-F or NMTF formulations become more challenging, since binary matrices usually pose multiple optimisation demands.

To address the above problems, in this chapter, we extend the standard NMF to learn binary features with a novel Bayesian model, termed infinite non-negative binary matrix tri-factorisation (iNBMT). In contrast to the traditional NMF, the novel iNBMT model is a statistical latent variable model which can automatically select an optimal feature set from infinite latent features, by applying the Indian Buffet Process (IBP) prior to factor matrices. Further, we decompose the input sample matrix $\mathbf{Y}$ into triple matrix factors, i.e., $\mathbf{Y} = \mathbf{Z}\mathbf{W}\mathbf{X}^T$, where $\mathbf{Z}$ and $\mathbf{X}$ are two binary matrices, and the non-negative matrix $\mathbf{W}$ can be considered a weight matrix. In comparison with bi-factorisation, which is typically involved in NMF, tri-factorisation can better capture latent features and reveal hidden structures underlying the samples [11]. For illustration, we plot our proposed novel iNBMT method as a graphic model in Fig. 4.1(b). Compared to the basic IBP based model which involves two factors (bi-factorisation) $\mathbf{Z}$ and $\mathbf{W}$ in Fig. 4.1(a), the proposed model employs one more binary matrix $\mathbf{X}$ with the IBP prior. The binary matrix $\mathbf{Z}$ can be considered to learn latent features, whilst the other binary matrix $\mathbf{X}$ extracts hidden structures of objects. Furthermore, the interrelation is leveraged by a non-negative weight matrix $\mathbf{W}$ which is also used to adjust the intensity of the features. Another well-known IBP-based method, termed correlated IBP-IBP (or IBP-IBP) is shown in Fig. 4.1(c). This model is also a tri-factorisation based method, which reveals relationships between categories and features by defining a category assignment matrix $\mathbf{U}$, and a set of category-feature relations $\mathbf{V}$. However, this approach requires a number of constraints and is consequently less flexible. More details are outlined in Section 4.4.

Approximation of the posterior in IBP is usually realised using Gibbs sampling. However, for large-scale matrices, variational methods can attain better performance than Gibbs sampling. This is due to the use of samplers in the latter, that often lead to mixing problems with growth in dimensionality [135]. Further, uncollapsed Gibbs samplers can easily get stuck in local optima. To circumvent this, we propose an efficient, modified variational-Bayes (VB) algorithm to fit the massive matrix decomposition, which can be thought of as a maximisation-expectation algorithm (ME-algorithm) [136]. More importantly, the time complexity of our proposed ME-algorithm is proved to be one order lower than other state-of-the-art models, such as the Maximisation-Expectation-IBP (ME-IBP) [20] and the IBP-IBP model [21].

There are a number of other related methods reported in the literature for learning latent binary features. For example, Zhang et al. extended the standard NMF to Bina-



(a) The basic model via IBP (b) iNBMT (c) IBP-IBP

Fig. 4.1: Comparison of different models via IBP. $\mathbf{Y}$ is the observed data, and $\mathbf{W}$ is the real matrix (or non-negative matrix in iNBMT and ME-IBP). $\mathbf{Z}$ and $\mathbf{X}$ are binary matrices (similar to $\mathbf{U}$ and $\mathbf{V}$ in IBP-IBP). $\theta, \alpha, \lambda, \mu, \sigma$ are fixed parameters. (a) ME-IBP: Bi-factorisation with a non-negative matrix and a binary matrix. (b) Our proposed iNBMF method: Tri-factorisation with a non-negative real matrix and two binary matrices. (c) IBP-IBP: Hierarchical structure includes a real matrix and two binary matrices (with constraints)

ry Matrix Factorisation (BMF) for producing biclustering structures [130]; the correlated IBP-IBP model [21] is also able to generate binary features. However, both these two models are limited. BMF requires input data to be strictly binary, which is too strong an assumption in real cases. The IBP-IBP model [21] enforces a product of two binary matrices to be binary, such assumption being invalid in general. Another recently-proposed ME-IBP model [20] is a bi-factorisation approach, which does not consider relational entities.

In summary, the contributions of this chapter can be outlined below:

- An IBP based infinite NMF model is proposed, which is able to automatically determine an optimal number of features.
- The proposed model offers tri-factorisation and is able to deliver latent binary features, addressing a more challenging optimisation problem.
- No extra constraints are enforced, presenting more appealing features when compared with other models used to generate binary features, such as the BMF model [130] and the IBP-IBP model [21].
- The iteration complexity is one order lower than competitive IBP-based models.
- The proposed model outperforms state-of-the-art methods (such as ME-IBP, correlated IBP-IBP, and PNMT) on both synthetic and real data for feature exaction, reconstruction, restoration and clustering.

The rest of this chapter is organised as follows. The Indian buffet process and maximisation-expectation algorithm are introduced as preliminaries in Section 4.1. A detailed description of our proposed iNBMT model is presented in Section 4.2, and the optimisation procedure is introduced in Section 4.3. In Section 4.4, related work is briefly reviewed. In particular, the difference between our proposed method and other relevant work is emphasised. In Section 4.5, the experimental setup is presented, including visualisation and quantitative results on seven datasets including a synthetic dataset, and six real-world datasets. A complexity analysis is carried out and compared with two related IBP-based methods in Section 4.6. Finally, concluding remarks, limitations and future work suggestions are presented in Section 4.7.

4.1 Background

This section presents an overview of latent feature learning through matrix factorisation, the infinite IBP prior and ME algorithm, as well as notations used throughout the chapter.

4.1.1 Learning Latent Feature via Matrix Factorisation

A latent feature model refers to a model where each object is associated with a subset of possible latent features. In this family of methods, each latent feature can influence the observations. Moreover, multiple latent features can be activated simultaneously. In a probabilistic latent feature model, the marginal distribution is written as follows

$$p(\mathbf{Y}) = \sum_{\mathbf{F}} p(\mathbf{Y}, \mathbf{F}) = \sum_{\mathbf{F}} p(\mathbf{Y}|\mathbf{F})p(\mathbf{F}) \,, \tag{4.1}$$

where $\mathbf{Y}$ represents real-valued observed objects, the matrix $\mathbf{F}$ is used to indicate latent features, $p(\mathbf{Y}|\mathbf{F})$ determines the probability of objects conditioned on these features, and $p(\mathbf{F})$ is the probability that objects are associated with each feature. In methods based on matrix factorisation, the latent features F are decomposed to $\mathbf{F} = \mathbf{Z} \bigotimes \mathbf{W}$, where $\bigotimes$ denotes element-wise product, $\mathbf{Z}$ determines which features are assigned to each object, and $\mathbf{W}$ indicates the value of each latent feature for each object. By specifying priors for $\mathbf{Z}$ and $\mathbf{W}$, the prior on $\mathbf{F}$ can be defined as $p(\mathbf{F}) = p(\mathbf{Z})p(\mathbf{W})$ when these two components are independent. Therefore, the focus becomes how to identify a prior on $\mathbf{Z}$, and hence determine the effective dimensionality of latent features. IBP is often used as the prior on infinite binary matrices. An IBP prior on $\mathbf{Z}$ can make the limit of feature numbers close to infinity. In the next subsection, we introduce the concepts of IBP as well as IBP prior.

4.1.2 Indian Buffet Process

IBP is termed a sequential process used to represent an infinitely exchangeable distribution of a stochastic process [137]. The process of IBP is detailed in the description of Figure 2.6 in Chapter 2. Further, the process reveals that IBP assumes an unbounded number of features, but the observed objects represent only a finite subset of features. IBP can be considered a prior for infinite binary matrices defined in these models. It is typically used to infer the number of latent features each observation possesses. We assume observed objects (N objects with D attributes) $\mathbf{Y} \in \mathbb{R}^{N\times D}$ are generated by linear combination of an assignment matrix $\mathbf{Z} \in \mathbb{R}^{N\times K}$ and a matrix $\mathbf{W} \in \mathbb{R}^{K\times D}$ (containing K latent factors). As a consequence, a latent feature model is portrayed as $\mathbf{Y} = \mathbf{ZW} + \epsilon$. ϵ is zero-mean, independently distributed Gaussian noise.

Assume an element of the binary matrix $z_{nk} = 1$ indicates that the object n has the latent feature k, $k = 1, \dots, K$. Each column of binary feature matrix $\mathbf{Z}$ is assumed to be over an IBP prior, derived independently by placing Beta priors from a Bernoulli distribution. The feature k is assigned to each object with probability π_k, which is generated independently from a Bernoulli distribution and the bias π_k is independently generated by a Beta prior, over each column[1]

$$\pi_k \mid \alpha \sim \textit{Beta}(\alpha/K, 1)\ , \quad \mathbf{Z} \mid \pi_k \sim \textit{Bernoulli}(\pi_k)\ ,$$
$$p([\mathbf{Z}]) = \prod_{k=1}^{K} \frac{\frac{\alpha}{K}\Gamma(m_k + \frac{\alpha}{K})\Gamma(N - m_k + 1)}{\Gamma(N + 1 + \frac{\alpha}{K})}\ . \tag{4.2}$$

Here Γ denotes the gamma function, α is the IBP strength parameter, and $m_k = \sum_{n=1}^{N} \mathbf{z}_{nk}$ is used to count the number of objects possessing feature k.

For infinite models, several classic matrix factorisation models have been developed as IBP inspired infinite-limit versions, for instance, infinite ICA models [138]. Intuitively, an infinite limit implies that the probability of $\mathbf{Z}$ will be specified in the infinite classes. To this end, Griffiths et al. made the number of attributes unbounded by proposing equivalence classes over binary matrices, in order to take the IBP prior into the infinite limit [72]. More importantly, since customers and dishes are exchangeable, following the principle of IBP, equivalence classes can be used to permutate the order of columns by eliminating all the null columns. Consequently, the infinite number of active features K_+ means K is unbounded, which is learned from data, while remaining finite with probability one. By rearranging the non-zero columns of $\mathbf{Z}$, we can specify $K \to \infty$ and modify Eq. (4.2) as follows

$$p([\mathbf{Z}]) = \frac{\alpha^{K_+}}{\prod_{h>0} K_h} e^{-\alpha H_N} \prod_{k=1}^{K_+} \frac{(N - m_k)!(m_k - 1)!}{N!}\ , \tag{4.3}$$

where $H_N = \sum_{j=1}^{N} \frac{1}{j}$ denotes the N^{th} harmonic number and K_h represents the number of non-zero rows. Moreover, K_h and m_k are both irrelevant to the objects sequence, which proves that $p([\mathbf{Z}])$ is an infinitely exchangeable distribution. In line with this property,

[1]A Beta prior is placed on π_k with the shape parameter α/K for all k and scale parameter 1.

IBP has been shown to be useful for binary factor analysis, such as modelling protein interactions, and similarity judgments [139, 140]. It has also been applied in other fields such as choice behaviour modelling, link prediction, and dictionary learning for correlated observations [141–144].

4.1.3 Maximisation-Expectation Algorithm

The variational Bayesian (VB) paradigm, as the basis of our proposed algorithm, has the ability to automatically select an optimal number of clusters from observations [145, 146]. The approximation process is an Expectation-Maximisation (EM)-like method, alternating between estimations of cluster assignments and stochastic parameters. Kurihara at al. further modified VB-clustering with fast implementation, termed the Maximisation-Expectation (ME) algorithm [136]. The ME algorithm simply reverses the roles of two steps in the classical EM algorithm, by maximising over hidden variables and marginalising over random parameters. We consider a probabilistic model $p(\mathbf{Y}, \mathbf{Z}, \mathbf{W})$ with observations $\mathbf{Y}$ and hidden random variables $\mathbf{Z}$ and $\mathbf{W}$. To carry out approximate Maximum-a-Posteriori (MAP) inference, it is necessary to compute posterior or marginal probabilities such as $p(\mathbf{Z}|\mathbf{Y})$, $p(\mathbf{W}|\mathbf{Y})$, or $p(\mathbf{Y})$. In Mean-Field Variational Bayes (MFVB) approximation, the posterior $p(\mathbf{Z}, \mathbf{W}|\mathbf{Y})$ cannot be computed analytically. Therefore, by assuming independent variational distributions, the posterior distribution can be factorised to variational distributions: $p(\mathbf{Z}, \mathbf{W}|\mathbf{Y}) \approx q(\mathbf{Z})q(\mathbf{W})$ [147]. These results are then iteratively updated as follows

$$\begin{aligned} &q(\mathbf{Z}) \propto \exp\left(\mathbb{E}_{q(\mathbf{W})}[\ln p(\mathbf{Z}, \mathbf{W} \mid \mathbf{Y})]\right) \\ &\leftrightarrow q(\mathbf{W}) \propto \exp\left(\mathbb{E}_{q(\mathbf{Z})}[\ln p(\mathbf{Z}, \mathbf{W}) \mid \mathbf{Y})]\right) . \end{aligned} \tag{4.4}$$

Here, the symbol $\leftrightarrow$ implies $q(\mathbf{Z})$ and $q(\mathbf{W})$ are updated iteratively. The VB approximation is based on minimising the Kullback-Leibler (KL)-divergence, which is shown as $KL[q(\mathbf{Z})q(\mathbf{W})\|p(\mathbf{Z}, \mathbf{W}|\mathbf{Y})]$. As a special case of MFVB, the ME algorithm maximises latent variables and then applies expectations over parameters. The results are close-formed with updates as follows

$$\begin{aligned} &q(\mathbf{W}) = p(\mathbf{W} \mid \mathbf{Z}^{*}, \mathbf{Y}) \\ &\leftrightarrow \mathbf{Z}^{*} = \arg\max_{\mathbf{Z}} \mathbb{E}_{q(\mathbf{W})}[\ln p(\mathbf{Z}, \mathbf{W} \mid \mathbf{Y})] . \end{aligned} \tag{4.5}$$

4.2 Infinite Non-negative Binary Matrix Tri-factorisation

In this section, we present our proposed iNBMT model which exploits IBP priors to associate an item and attributes with more than one cluster. We first describe the model

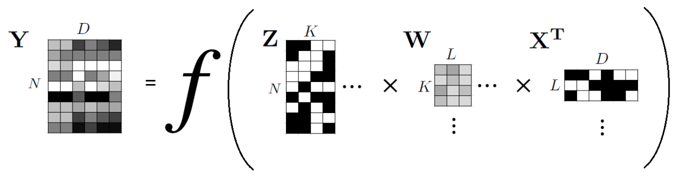


Fig. 4.2: Representation of the iNBMT model. The process $f\ (\cdot)$ applied to the linear inner product of the three components. Here $\mathbf{Z}$ and $\mathbf{X}$ are infinite binary matrices, $\mathbf{W}$ represents a non-negative matrix.

and its formulation, and then show how to employ a modified, efficient ME-algorithm to perform approximate MAP inference.

4.2.1 Model Description

The iNBMT model is applied to real-valued observation data $\mathbf{Y} \in \mathbb{R}^{N\times D}$ with exchangeable rows and columns. Considering a probabilistic latent feature model in Eq. (4.1), our focus is on latent features $p(\mathbf{F})$, the difference being that these are further decomposed into three components: $\mathbf{F} = \mathbf{Z} \bigotimes \mathbf{W} \bigotimes \mathbf{X}$. In this feature-based model, for the case of L latent features, $\mathbf{X}$ is an $L \times D$ binary feature matrix. Furthermore, the K^{th} potential binary vector $\mathbf{z}_i$ denotes the feature vector corresponding to entity i, and the $K \times L$ interactive weight matrix $\mathbf{W}$ represents the primary parameters. Assuming the three components of $\mathbf{F}$ are independent, the priors of the features are defined by: $p(\mathbf{F}) = p(\mathbf{Z})p(\mathbf{W})p(\mathbf{X})$. In Fig. 4.2, the iNBMT model is illustrated pictorially, where the observations $\mathbf{Y}$ are represented by $\mathbf{ZWX^T}$ depending on a fixed observation distribution $f\ (\cdot)$

$$\mathbf{Y} \mid \mathbf{Z}, \mathbf{W}, \mathbf{X} \sim f\,(\mathbf{ZWX}^T, \boldsymbol{\theta}) \ , \tag{4.6}$$

where $\boldsymbol{\theta}$ are hyperparameters specific to the model variant. This process is equivalent to factorisation or approximation of the data.

Our focus is to learn the latent features automatically by placing Bayesian non-parametric priors on binary matrices. Unlike the matrix factorisation method, our tri-factorisation method does not need to place an upper bound on the number of features, or on the number of clusters. In infinite models, both the binary matrices $\mathbf{Z}$ and $\mathbf{X}$ are assumed to be matrices with an unbounded number of columns. Specifically, the IBP priors are imposed over infinite binary matrices with the property that, a finite number of entities will only have a finite number of non-zero features, with a probability of 1. Thus, the binary matrix always has a positive probability under the IBP prior. Our basic generative model can then be shown as below

$$\mathbf{Z} \sim \mathcal{IBP}(\alpha) \ , \quad \mathbf{X} \sim \mathcal{IBP}(\lambda) \ , \quad \mathbf{W} \sim \mathcal{F}(\mathbf{W}; \boldsymbol{\mu}, \boldsymbol{\sigma}_W^2) \ . \tag{4.7}$$

Here, any non-negative prior $\mathcal{F}$ (e.g., exponential and truncated Gaussian) is assumed on the weight matrix $\mathbf{W}$. The IBP prior is stated in Eq. (4.2), and the hyperparameters $\boldsymbol{\theta}$ conjugate gamma priors on inference parameters. In our iNBMT model, each object possesses multiple latent features, and each latent feature is also assigned to numerous latent classes by using two IBP priors. Moreover, both latent features and latent classes are associated with a distribution over attributes. Thus, our proposed method can be used for discovering fundamental hidden structures in complex data.

4.2.2 Linear-Gaussian iNBMT Model: Formulation

In this section, we derive linear-Gaussian as an observation distribution, with mean $\mathbf{ZWX}^T$ and covariance $(1/\boldsymbol{\theta})\mathbf{I}$ for capturing the latent features.

The Gaussian distribution of $\mathbf{Y}$ given $\mathbf{A} = \{\mathbf{Z}, \mathbf{W}, \mathbf{X}\}$ and $\boldsymbol{\sigma}_Y^2$ is shown as below

$$p(\mathbf{Y}|\mathbf{A}, \boldsymbol{\sigma}_Y^2) = \frac{1}{(2\pi\boldsymbol{\sigma}_Y^2)^{\frac{ND}{2}}} \exp \frac{-tr\big((\mathbf{Y} - \mathbb{E}[\mathbf{Y}])^T(\mathbf{Y} - \mathbb{E}[\mathbf{Y}])\big)}{2\boldsymbol{\sigma}_Y^2} , \tag{4.8}$$

where $\mathbb{E}[\mathbf{Y}] = \mathbf{ZWX}^T$. The linear-Gaussian iNBMT model can be considered a two-sided version of the linear-Gaussian model. The truncated Gaussian (*TN*) prior is placed on the non-negative interactive matrix $\mathbf{W}$, with mean zero and covariance matrix $\boldsymbol{\sigma}_W^2$

$$p(\mathbf{W}|0, \boldsymbol{\sigma}_W^2) = \prod_{k=1}^{K}\prod_{l=1}^{L} TN(\mathbf{w}_{kl}; 0, \boldsymbol{\sigma}_W^2) . \tag{4.9}$$

According to Eq. (4.3), the marginal probabilities $p([\mathbf{Z}])$ and $p([\mathbf{X}])$ are specified with the infinite IBP prior

$$p(\mathbf{Z}|\alpha) = \frac{\alpha^{K_+}}{K_+!} e^{-\alpha H_N} \prod_{k=1}^{K_+} \frac{(N - m_k)!(m_k - 1)!}{N!} . \tag{4.10}$$

Here, $m_k = \sum_{n=1}^{N} z_{nk}$, the $p(\mathbf{X}|\lambda)$ follows the same formula structure with $N \leftarrow D$, $k \leftarrow l$ $(l = 1, \ldots, L)$ and $m_k \leftarrow s_l = \sum_{d=1}^{D} x_{dl}$ ($\leftarrow$ denotes a substitution).

From the Bayesian theorem, the likelihood can be written as follows

$$p(\mathbf{Y}, \mathbf{A}|\boldsymbol{\theta}) = p(\mathbf{Y}|\mathbf{A}, \boldsymbol{\sigma}_Y^2)p(\mathbf{W}|0, \boldsymbol{\sigma}_W^2)p(\mathbf{Z}|\alpha)p(\mathbf{X}|\lambda) . \tag{4.11}$$

We assume the hyperparameters $\boldsymbol{\theta} = \{\alpha, \lambda, \sigma_Y, \sigma_W\}$ are estimated from the data. By placing conjugate gamma hyper-priors on these parameters, we can employ a straightforward extension to infer their values [146]. Formally, [2]

$$\begin{aligned} &p(\alpha) \sim \mathcal{G}(a_\alpha , b_\alpha), \;\; p(\lambda) \sim \mathcal{G}(a_\lambda, b_\lambda) , \\ &p(\sigma_Y) \sim \mathcal{IG}(a_{\sigma_Y}, b_{\sigma_Y}) , \;\; p(\sigma_W) \sim \mathcal{IG}(a_{\sigma_W}, b_{\sigma_W}) . \end{aligned} \tag{4.12}$$

In the above, $\mathcal{G}$ denotes the Gamma prior, and $\mathcal{IG}$ refers to the inverse Gamma prior.

[2]A random variable X follows a gamma-distributed, $X \sim \Gamma(\alpha, \beta) \equiv \text{Gamma}(\alpha, \beta)$, where α is shape parameter and β is rate parameter

4.2.3 Linear-Gaussian iNBMT Model: Variational Inference Procedure

Here, the variational inference procedure is presented for the linear-Gaussian iNBMT model. Consider a model with observations $\mathbf{Y}$, hidden variables $\mathbf{A} = \{\mathbf{Z}, \mathbf{W}, \mathbf{X}\}$, and hyperparameters $\boldsymbol{\theta}$. In the optimisation stage, these variables often work with the log-marginal likelihood of the observations

$$\log p(\mathbf{Y}|\boldsymbol{\theta}) = \log \int p(\mathbf{Y}|\mathbf{A}, \boldsymbol{\theta}) d\mathbf{A} \ . \tag{4.13}$$

However, the log-marginal probability is difficult to compute, which implies the true log-posterior calculation is also intractable. In order to approximate the true posterior, the mean field variational method is developed with a variational distribution $q_\nu(\mathbf{A})$ (where ν is the variational parameter). The inference is then applied to the variational distribution, by optimising the KL divergence. In particular, the aim is to minimise the KL divergence $\mathbf{D}(q \parallel p)$ between $q_\nu(\mathbf{A})$ and $p(\mathbf{A}|\mathbf{Y}, \boldsymbol{\theta})$

$$\mathbf{D}(q \parallel p) = \log p(\mathbf{Y}|\boldsymbol{\theta}) + \mathbb{E}_q[\log q_\nu(\mathbf{A})] - \mathbb{E}_q[\log p(\mathbf{Y}, \mathbf{A}|\boldsymbol{\theta})] \ . \tag{4.14}$$

Alternatively, this is equivalent to maximising a lower bound on the log-marginal likelihood

$$\begin{aligned} \ln p(\mathbf{Y}|\boldsymbol{\theta}) =& \mathbb{E}_q[\log p(\mathbf{Y}, \mathbf{A}|\boldsymbol{\theta})] - \mathbb{E}_q[\log q_\nu(\mathbf{A})] + \mathbf{D}(q \parallel p) \\ \geq& \mathbb{E}_q[\ln p(\mathbf{Y}, \mathbf{Z}, \mathbf{W}, \mathbf{X}|\boldsymbol{\theta})] + \mathcal{H}[q] \equiv \mathcal{T} \ , \end{aligned} \tag{4.15}$$

where $\mathcal{H}[q]$ is the entropy of q. Importantly, the approximate MAP inference is derived from the ME framework by following Eq. (4.5), i.e., maximising the latent variables and taking expectations over variational parameters. In the linear-Gaussian iNBMT model, the lower bound of evidence, $\mathcal{T}$, is written as follows

$$\begin{aligned} \mathcal{T} \equiv& \frac{1}{\sigma_Y^2}\big[- \frac{1}{2}(\mathbf{Z}\mathbb{E}[\mathbf{W}]\mathbf{X}^T)(\mathbf{Z}\mathbb{E}[\mathbf{W}]\mathbf{X}^T)^T + \mathbf{Z}(\mathbb{E}[\mathbf{W}]\mathbf{Y}^T + \mathbf{Z}\gamma)\mathbf{X}^T\big] \\ & + \log p(\mathbf{Z}|\alpha) + \log p(\mathbf{X}|\lambda) + \sum_{k=1}^{K_+}\sum_{l=1}^{L_+} \varphi_{kl} + const \ , \end{aligned} \tag{4.16}$$

with

$$\begin{aligned} \gamma &= \frac{1}{2}\sum_{k=1}^{K}\sum_{l=1}^{L} \big[\mathbb{E}[w_{kl}]^2 - \mathbb{E}[w_{kl}^2]\big]^T \ , \\ \varphi_{kl} &= -\frac{KL}{2}\ln\Big(\frac{\pi\boldsymbol{\sigma}_W^2}{2}\Big) - \frac{\mathbb{E}[w_{kl}^2]}{2\boldsymbol{\sigma}_W^2} + \mathcal{H}\big(q(w_{kl})\big) \ . \end{aligned}$$

Here $\mathbb{E}[\mathbf{W}]$ is a matrix with each element defined as $\mathbb{E}[w_{kl}]$. The variational parameters updating is described in the next subsection.

4.2.4 Linear-Gaussian iNBMT Model: Parameter Updating

Our proposed modified ME-algorithm adopts VB evidence to iteratively optimise a lower bound on the model marginal likelihood. This optimisation algorithm is particularly tailored to tri-factorisation, which has not been exploited previously in the literature. Specifically, the major improvement over the previous ME algorithm is that we design two loops in each iteration with a guaranteed convergence, which enables the algorithm to update two different binary matrices. Note that the previous ME algorithm can merely be used for bi-factorisation. In addition, compared with other popular Gibbs-sampler optimisation algorithms [72], our proposed algorithm engages VB evidence for inference. The latter is theoretically more rigorous than Gibbs-sampler based algorithms and usually leads to better performance [135]. Algorithm 3 enumerates the steps of our proposed modified ME algorithm. Specifically, parameters associated with the two infinite variables $\mathbf{Z}$ and $\mathbf{X}$ are updated, in turn. For completeness, these VB update equations are provided in Section 4.3.

Algorithm 3: Parameter Updating

```
Initial:
Give upper bounds of $K$ and $L$.
Generate the $N \times K$ binary matrix $\mathbf{Z}$, $L \times D$ binary matrix $\mathbf{X}$ and $K \times L$
non-negative matrix $\mathbf{W}$ randomly.
Repeat
    for $i = 1$ to $N$ do
        Optimise the binary matrix $\mathbf{Z}$ (in Section 4.3) by reducing the number of $K$
        with left-order form.
        Resize and update the matrix $\mathbf{W}$ based on the resized $\mathbf{Z}$ (in Section 4.3.1).
    for $j = 1$ to $D$ do
        Optimise the binary matrix $\mathbf{X}$ (in Section 4.3) by reducing the number of $L$
        with left-order form.
        Resize and update the matrix $\mathbf{W}$ based on the resized $\mathbf{X}$ (in Section 4.3.1).
    Calculate the Log-likelihood LogL.
Until $|\text{LogL}_{new} - \text{LogL}_{old}| <$ threshold value.
```

4.3 Optimising Latent Features

We introduce latent features optimisation in this subsection. Updating $\mathbf{Z}$ and $\mathbf{X}$ is relatively straightforward by computing Eq. (4.16). Similar to variational IBP methods, we split the expectation in Eq. (4.5) into terms depending on each of the latent variables [135], with the benefit that binary variables updates are not affected by inactive features. Thus, we decompose the relevant terms of $\mathbf{X}$ in Eq. (4.15). Similarly, we also decompose the terms depending on $\mathbf{Z}$ during updating. First, to decompose $\ln \frac{(D-s_l)!(s_l-1)!}{D!}$, we define a quadratic pseudo-Boolean function

$$f(x_{dl}) = \begin{cases} 0, \ if \ s_{l\backslash d} = 0 \ and \ x_{dl} = 0 \ ; \\ \ln \frac{(D-s_{l\backslash d}-x_{dl})!(s_{l\backslash d}+x_{dl}-1)!}{D!} \ , \text{ otherwise} \ . \end{cases}$$

Here $s_{l\backslash d}$ indicates the sum of x_{dl} after removing the d^{th} row from $\mathbf{X}$. The terms of Eq. (4.16), $\sum^{L_+}[\ln \frac{(D-s_l)!(s_l-1)!}{D!}] + \ln L_+!$, are changed to

$$\begin{aligned} &\sum_{l=1}^{L_+} f(x_{dl}) = \sum_{l=1}^{L_+} x_{dl}\big(f(x_{dl}=1) - f(x_{dl}=0)\big) + f(x_{dl}=0) \ , \\ &\ln L_+! = \ln\big(L_{+\backslash d} + \sum_{l=1}^{L_+}[\mathbf{1}_{\{s_{l\backslash d}=0\}} x_{dl}]\big)! \ , \end{aligned} \tag{4.17}$$

where $\mathbf{1}_{\{\cdot\}}$ is the indicator function. Here we show that the lower bound of the evidence Eq. (4.15) is well defined in the limit $L \to \infty$

$$\begin{aligned} \mathcal{T}(\mathbf{X}_{dl}) = &-\frac{1}{2\sigma_{\mathbf{Y}}^2}(\mathbf{X_{dl}}\mathbf{\Lambda}_{nl}^T)(\mathbf{X}_{dl}\mathbf{\Lambda}_{nl}^T)^T + \mathbf{X}_{dl}\boldsymbol{\omega}_{dl}^T + \sum_{l=1}^{L_+} f(x_{dl}) \\ &- \ln L_+! + \ln p(\mathbf{Z}|\alpha) + \sum_{k=1}^{K}(\mathbf{1}_{\{s_{l\backslash d}=0\}} x_{dl}\boldsymbol{\varphi}_{kl}) + const \ , \end{aligned} \tag{4.18}$$

where $\omega_{dl} = -\frac{1}{\sigma_{\mathbf{Y}}^2}(\mathbf{\Lambda}_{nl}^T \mathbf{Y}_{nd} + \boldsymbol{\gamma})$ and $\mathbf{\Lambda}_{nl} = \mathbf{Z}\mathbb{E}[\mathbf{W}]$.

4.3.1 Updating Variational Parameters

The updating of variational parameters for the non-negative matrix $\mathbf{W}$, over a truncated Gaussian distribution, is outlined below

$$q(\mathbf{W}) = \prod_{k=1}^{K}\prod_{l=1}^{L} TN(w_{kl}; \mu_{kl}, \sigma_{kl}^2) = \prod_{k=1}^{K}\prod_{l=1}^{L} \frac{N(\mu_{kl}, \sigma_{kl}^2)}{\mathbf{\Phi}(\infty) - \mathbf{\Phi}(0)} \ , \tag{4.19}$$

where $\mathbf{\Phi}(a) = \frac{1}{2}(1 + erf(\frac{a-\mu_{kl}}{\sqrt{2}\sigma_{kl}}))$, $\mathbf{\Phi}(\infty) = 1$ and $erf(\cdot)$ is the Gaussian error function.

In accordance with the upper tail truncation, the parameters are updated as follows

$$\mathbb{E}[w_{kl}] = \mu_{kl} + \sigma_{kl}\lambda(t) \ , \quad \mathbb{E}[w_{kl}^2] = \mu_{kl}\mathbb{E}[w_{kl}] + \sigma_{kl}^2 \ , \tag{4.20}$$

with $\lambda(t) = \sqrt{2}/\sqrt{\pi}e^{t^2}(1 - erf(t))$, $t = -\mu_{kl}/\sqrt{2}\sigma_{kl}$. It is worth noting that the mean and variance of truncated Gaussian distributions need be computed twice per iteration

$$\begin{aligned}
if\ K \to \infty,\ \mu_{k_+l} &= \tau^2 \sum_{n=1}^{N} z_{nk}^T \big(y_{nd} - \sum_{k'/k} z_{nk'}\mathbb{E}[w_{k'l}x_{dl}^T]\big)x_{dl}\ , \\
\sigma_{k_+l} &= \tau\sigma_Y\ ; \\
if\ L \to \infty,\ \mu_{kl_+} &= \tau^2 \sum_{d=1}^{D} x_{dl} \big(y_{nd}^T - \sum_{l'/l} x_{dl}^T\mathbb{E}[w_{kl'}z_{nk'}]\big)z_{nk}^T\ , \\
\sigma_{kl_+} &= \tau\sigma_Y\ ,
\end{aligned} \tag{4.21}$$

where $\tau = (m_k^T s_l + \frac{\sigma_Y^2}{\sigma_W^2})^{-\frac{1}{2}}$. Finally, the entropy of truncated Gaussian distribution is given as

$$\begin{aligned}
\mathcal{H}(q(w_{kl})) =& \frac{1}{2\sigma_{kl}^2}\big\{\mathbb{E}[w_{kl}]^2 - \mathbb{E}[w_{kl}^2] - (\mathbb{E}[w_{kl}] - \mu_{kl})^2 \\
&- \big[\frac{1}{2}\ln\frac{2}{\pi\sigma_{kl}^2} - \ln(1 - erf(t))\big]\big\}\ .
\end{aligned} \tag{4.22}$$

4.4 Benchmarking Approaches

IBP is frequently used by computational intelligence and machine learning communities. In particular, several feature-based models have been reported that exploit the IBP [138, 148]. In the following, we mainly introduce factorisation methods utilising IBP, that are closely related to our proposed model.

The first benchmark, infinite latent-feature model based on IBP priors, was proposed by Griffiths et al. [149]. Reed et al. introduced the linear-Gaussian IBP model by employing a maximisation-expectation framework to approximate the MAP inference (ME-IBP) [20, 136]. This particular model can be regarded as a latent factor model in which the binary matrix $\mathbf{Z}$ linearly combines the latent factors $\mathbf{W}$. The graphic structure of ME-IBP can be seen in Fig. 4.1(a) (detailed formulations are shown in Chaper 2.3.2). As can be seen, the ME-IBP model draws $\mathbf{Z}$ from an IBP prior, with the strength parameter α_z. The full IBP model is shown in Eq. (4.2). The prior over the non-negative matrix $\mathbf{W}$ is an independent, identically-distributed, truncated-Gaussian with zero mean and covariance $\boldsymbol{\sigma}_W^2$. Further, $\mathbf{Z}$ can be optimised by maximising a submodule cost function

$$\mathbf{Z} \sim \mathcal{IBP}(\alpha_z)\ , \quad \mathbf{W} \sim \mathcal{T}(\mathbf{W}; 0, \boldsymbol{\sigma}_W^2)\ .$$

The ME-IBP is known to be significantly slower than many other methods reported in the literature, and is apparently unable to learn binary features in bi-factorisation settings.

Some researchers [150, 151] have attempted to extend the basic IBP model, by considering the correlation between latent features and observations. In particular, the correlated IBP-IBP model employs a correlation framework for feature-based models [21].

This model is perhaps the most relevant for benchmarking our proposed method. The tri-factorisation of this model is illustrated using a hierarchical structure in Fig. 4.1(c). As can be seen from this hierarchical structure, the feature assignments matrix $\mathbf{Z}$ is further decomposed into two binary matrices. Further, in this IBP-IBP model, the IBP prior is drawn on both binary variables, and the Bernoulli function is set as the link function. The latent feature matrix $\mathbf{W}$ follows an exponential prior, as below

$$\mathbf{U} \sim \mathcal{IBP}(\alpha_u) \ , \mathbf{V} \sim \mathcal{IBP}(\alpha_v) \ , \mathbf{z_{nk}} \sim Bernoulli(1 - q^{u_n^T v_k}) \ .$$

Here, α_u, and α_v are the strength parameters, and $q \in [0, 1]$ is the noise parameter. In the IBP-IBP model, u_n is the n^{th} row of the binary category matrix $\mathbf{U}$, and v_k is the k^{th} column of the binary matrix $\mathbf{V}$, which denotes the category-feature relations. $\mathbf{z}_{nk} = 1$ means the feature k is presented in observation n. It is worth noting that, $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{Z}$ are all considered as the binary matrix, forcing observations to be associated with only one category. This constraint limits the model's flexibility. Thus, the weakness of the IBP-IBP model is on identifiability issues commonly associated with feature-based models.

There are also other NMTF methods that do not adopt IBP. These models are usually based on a discriminative model [11] aiming to minimise the following objective

$$\arg\min \parallel \mathbf{Y} - \mathbf{Z}\mathbf{W}\mathbf{X}^T \parallel_2 \ , \quad s.t. \quad \mathbf{Z} \in \mathbb{R}_+^{D \times l}, \mathbf{X} \in \mathbb{R}_+^{N \times k} \ .$$

Here, $\mathbf{Y}$ is a $D \times N$ observation matrix; l and k represent the number of object clusters and feature clusters separately. For addressing co-clustering, this method was further developed by constraining the factors $\mathbf{Z}$ and $\mathbf{X}$ [131]. More recently, PNMT further decomposed each of the factors with three penalty-term constraints [132].

4.5 Experiments

In this section, we experimentally evaluate our proposed iNBMT model on seven dataset-s, five of which are used for tasks of feature extraction, reconstruction and pre-image restoration. We compare the performance of iNBMT with three state-of-art algorithm-s: Maximisation-Expectation-IBP (ME-IBP), correlated IBP-IBP (IBP-IBP) and Penalty NMTF (PNMT). In addition, the clustering performance of our proposed method is e-valuated on two benchmark datasets and compared with four related methods, including K-means, ME-IBP, PNMT, and Fast NMTF (FNMTF).

4.5.1 Datasets

We first evaluate our method on a synthetic dataset and then conduct experiments on six real-world datasets obtained from real tasks. For the synthetic data, we modify the one

Table 4.1: Description of seven datasets used in the experiment

Datasets	Noise term	Training	Testing	Dimensions	Classes
Syn	$\sigma_Y = 0.8$	4500	-	36	-
Com-USPS	$\sigma_Y = 0.8$	2000	-	1024	-
Com-NIST	$\sigma_Y = 0.8$	400	-	4096	-
Pre-USPS	-	400	20	256	-
Pre-NIST	-	600	15	1024	-
Coil-20	-	1440	-	4096	20
UMIST	-	575	-	644	20

used in Griffiths [72]. For the remaining six datasets, Com-USPS, Pre-USPS, Com-NIST, and Pre-NIST are reused from the well-known USPS and NIST datasets respectively, whereas Coil-20 and UMIST are two widely-used benchmark clustering datasets. Information on these seven datasets is given in Table 4.1, where $\mathbf{Y}$ denotes the observations and σ_Y indicates the variance of the Gaussian noise. The first three datasets, i.e., Syn, Com-USPS, and Com-NIST, are used for evaluating the performance of different methods for feature extraction and reconstruction tasks, while the Pre-USPS and Pre-NIST are used to validate various models for the pre-image restoration task. All the five datasets are available online at: `https://github.com/zzy8989/Data-iNBMT`. Finally, the Coil-20 and UMIST are used to evaluate the clustering performance.

Synthetic Data

The synthetic dataset was generated by modifying the dataset used in Griffiths [72]. Specifically, our dataset comprises 6×6 grey images adapted via three different luminance levels, as illustrated in Fig. 4.6(a). Each row of the observations $\mathbf{Y}$ is generated using $\mathbf{Z}$ to linearly combine a subset of four binary factors $\mathbf{X}$ (see Fig. 4.3(a)). In addition, $\mathbf{W}$ loads different luminance combinations. The modified dataset presents a more challenging problem and appears more appropriate for evaluating the different methods.

Com-USPS and Pre-USPS Data

We generated two datasets from USPS: the Com-USPS and the Pre-USPS. The digits, used in these two datasets, are sampled randomly from the USPS. Moreover, our generated datasets are scaled to $[0, 1]$. Each row of the Com-USPS dataset is built up with 32×32 grey images. Various kinds of digits $0, 1, 2, 3$ are combined with each sample, as illustrated in Fig. 4.7(a). The Pre-USPS dataset contains merely a single handwritten digit, which is also chosen randomly from $0, 1, 2, 3$. In the training set, each digit has 100

samples. Furthermore, in order to see if the various methods can restore an image, the test samples are bottom-halved from the original images (see Fig. 4.10(b)).

Com-NIST and Pre-NIST Data

The rest two datasets were both generated from NIST handprinted forms and characters database. It is worth mentioning that all the images are binary in this database. In the way same as generating Com-USPS, we generated the Com-NIST dataset so that each sample of the Com-NIST consists of 64×64 binary images combined from letters a, b, c, d (see Fig. 4.5(a)). On the other hand, in the Pre-NIST dataset, each sample contains a single handwritten letter chosen randomly from a, c, d. In the training set, each letter has 200 samples. Similarly, the test samples are top-halved from the original images so as to validate if the various methods can restore them (see Fig. 4.11(b)).

Coil-20-product and UMIST

Coil-20-product is an object recognition benchmark dataset consisting of grayscaling images from 20 objects. These objects have diversified reflection properties and complex geometric. Each object was rotated 360 degrees by the turntable, and 72 images were taken per object (rotated once every 5 degrees) [108].

UMIST is a face recognition benchmark dataset containing 575 images from 20 different subjects [152]. Each subject is with the different view. Each image is rescaled to 28×23.[3]

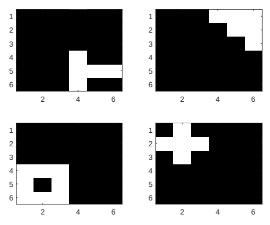
4.5.2 Feature Extraction

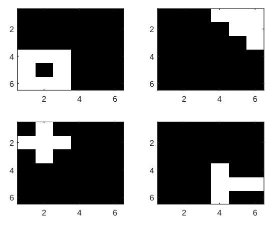
The three datasets Syn, Com-USPS, and Com-NIST are used to evaluate if different methods can extract the latent features.

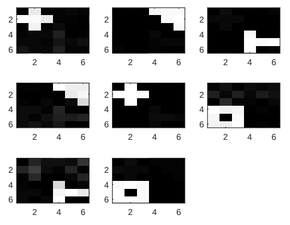
Fig. 4.3 shows the results on the Syn dataset. Fig. 4.3(a) provides the ground-truth latent features. Fig. 4.3(c) shows the inferred features by the ME-IBP, which matches well the truth features. It is however noted that each feature is repeated twice with certain noises. Compared with the ME-IBP, the learning features of the IBP-IBP are shown in Fig. 4.3(d), where the features are also repeated. In Fig. 4.3(e), the inferred features by the PNMT match the truth features, but with a lot of noise. In Fig. 4.3(b), it is evident that the iNBMT outperforms the above three. It perfectly matches the truth features as well as identifying the feature number automatically.

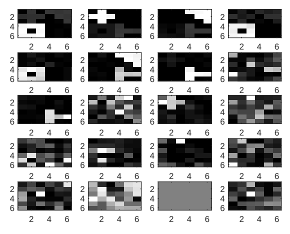
We next report the performance on the Com-USPS. The results are shown in Fig. 4.4. We presented the 9 input images as examples without noise for comparison (see Fig. 4.4(a)).

[3] https://www.sheffield.ac.uk/eee/research/iel/research/face


(a) Ground-truth latent features


(b) Features learned by iNBMT


(c) Features learned by ME-IBP


(d) Features learned by IBP-IBP

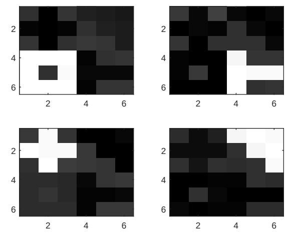
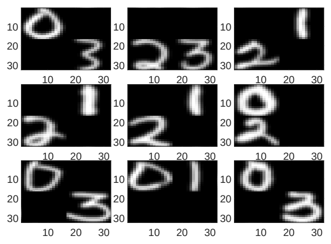
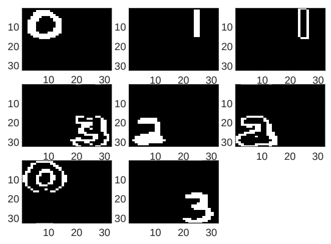
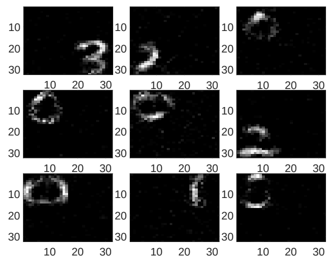
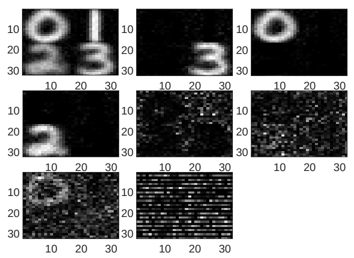

(e) Features learned by PNMT

Fig. 4.3: Comparison of iNBMT, ME-IBP, IBP-IBP, and PNMT on the synthetic dataset. iNBMT perfectly matches the truth features.


(a) 9 examples of training data without noise


(b) Features learned by iNBMT


(c) Features learned by ME-IBP


(d) Features learned by IBP-IBP

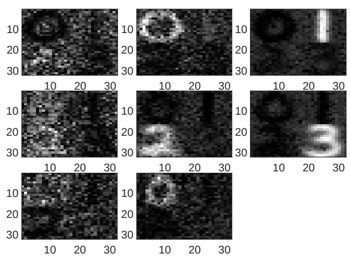
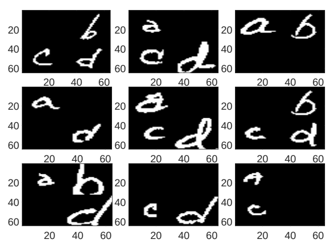
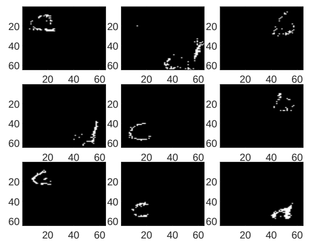
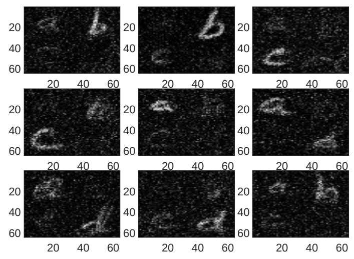
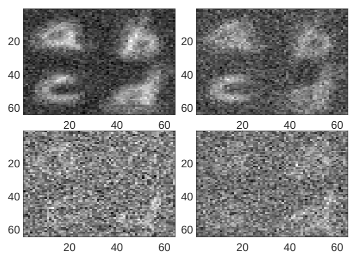

(e) Features learned by PNMT

Fig. 4.4: Comparison of iNBMT, ME-IBP, IBP-IBP, and PNMT on Com-USPS dataset. The first sub-figure shows 9 examples of training data. The other four show the features learned by each method. iNBMT clearly shows the best performance.


(a) 9 examples of training data without noise


(b) Features learned by iNBMT


(c) Features learned by ME-IBP


(d) Features learned by IBP-IBP

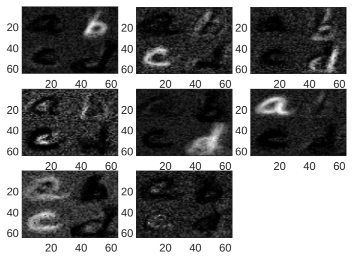

(e) Features learned by PNMT

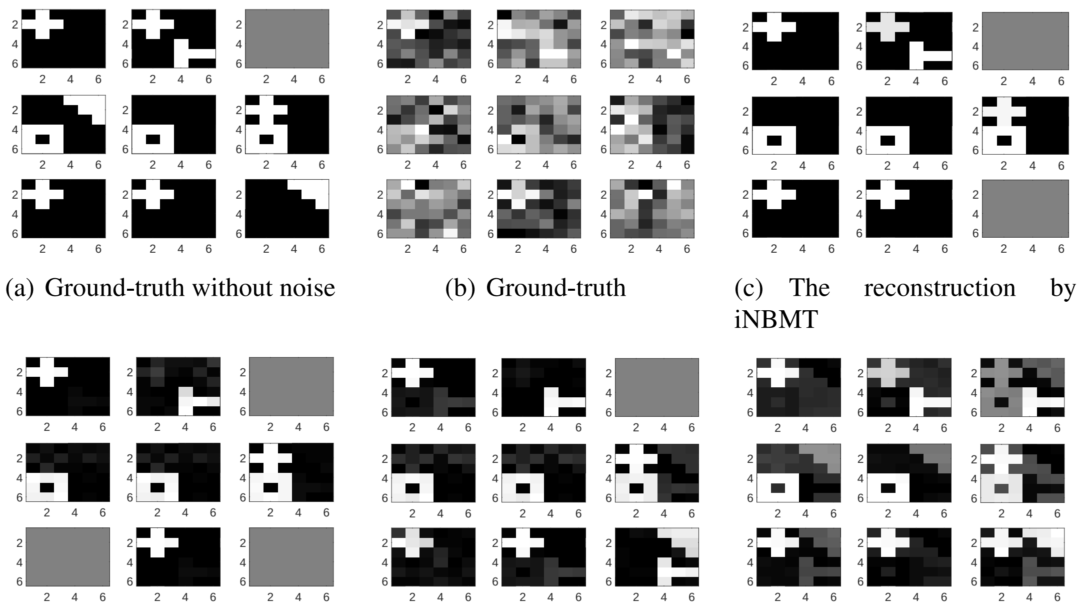
Fig. 4.5: Comparison of iNBMT, ME-IBP, IBP-IBP, and PNMT on Com-NIST dataset. The first sub-figure shows 9 examples of training data and the others show the features learned by each method. iNBMT shows the best performance.

It is interesting to see from Fig. 4.4(b), our proposed iNBMT not only captures the latent features, i.e., each of the clean digits, but also captures their image contours. In Fig. 4.4(e), PNMT also captures the single digits, but the learned features have more noise. On the other hand, the inferred features of the ME-IBP seem good, but these features are repeated many times, as shown in Fig. 4.4(c). In Fig. 4.4(d), the learned features of the IBP-IBP are provided. Apparently, its performance is not as good as our proposed iNBMT, and it is also incapable of obtaining the feature of digit 1.

We then present the evaluation results on the Com-NIST data in Fig. 4.5. It is evident that none algorithm can filter noise perfectly. Nonetheless, iNBMT still learned the underlying feature (the single letter) as clearly observed in Fig. 4.5(b). The ME-IBP and PNMT also extracted clear letters, but their results contained merely combinations of letters (see Fig. 4.5(c), Fig. 4.5(e)). It is usually tricky to cover all the possible combinations. Consequently, its performance is often much worse than the proposed iNBMT model. In summary, from the three groups of results, iNBMT demonstrates the best performance and outperforms the ME-IBP, IBP-IBP, and PNMT on feature extraction.

4.5.3 Reconstruction

In this section, we report experiments to compare the reconstruction performance of different methods.



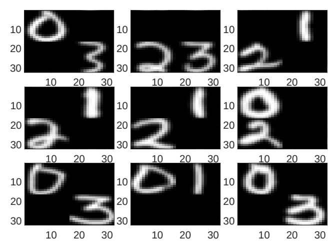
(a) Ground-truth without noise (b) Ground-truth (c) The reconstruction by iNBMT

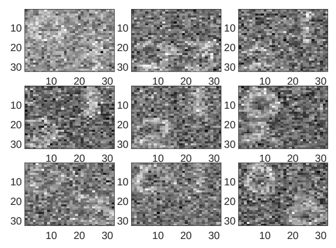
(d) The reconstruction by ME-IBP (e) The reconstruction by IBP-IBP (f) The reconstruction by PN-MT

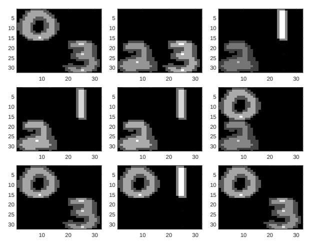
Fig. 4.6: Comparison of reconstruction of synthetic data. iNBMT best matches the groundtruth than ME-IBP, IBP-IBP, and PNMT.

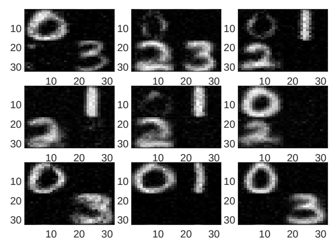
In the experimental setup, all three groups of observations are corrupted with $\sigma_Y = 0.8$ Gaussian noise. Examples of randomly generated images and their corrupted versions are illustrated in (a) and (b) respectively, of each figure (Fig. 4.6, Fig. 4.7, Fig. 4.8). The reconstructed images by four algorithms are shown in (c), (d), (e), (f) respectively, in each figure. We can clearly see that the images reconstructed from the iNBMT are more similar to the ground-truth. In particular, on the Com-USPS dataset, the iNBMT almost perfectly recovers the images. Significantly, the iNBMT denoising ability is also superior to that of other algorithms. Although several repeated features are extracted by iNBMT in the Com-NIST data (as seen in the feature extraction section), it does not prevent the iNBMT from producing reasonably good reconstructions of the data. In comparison, ME-IBP, IBP-IBP, and PNMT cannot clearly extract single digits or letters, on account of the latent features, and their reconstruction results being worse than those of the iNBMT. Moreover, exploiting the iNBMT framework, $\mathbf{W} \times \mathbf{X}^T$ can be considered as a set of basis images which can be added together with binary coefficients $\mathbf{Z}$ to recover images. It is apparent that all digit combinations are correctly detected. By adjusting features that are non-zero in each row of $\mathbf{Z}$, reconstructed images are composed by adding basis images together.

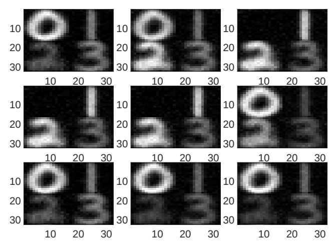
In order to quantitatively evaluate the reconstruction performance of different algorithms, we exploit Von-Neumann divergence as a criterion to measure the similarity be-


(a) Ground-truth without noise


(b) Corrupted ground-truth


(c) Reconstruction by iNBMT


(d) Reconstruction by ME-IBP


(e) Reconstruction by IBP-IBP

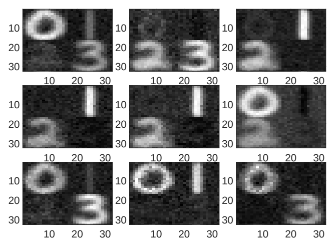
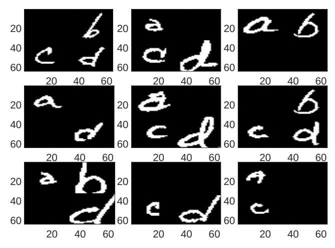
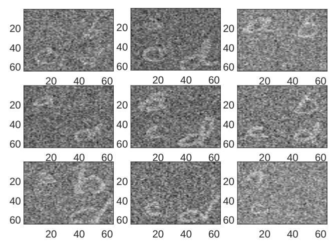
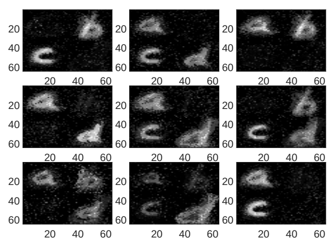
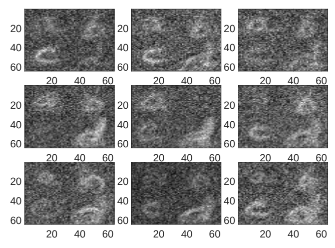

(f) Reconstruction by PNMT

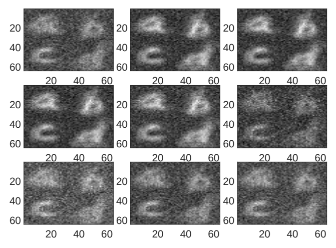
Fig. 4.7: Comparison of iNBMT, ME-IBP, IBP-IBP, and PNMT on the Com-USPS dataset. The first sub-figure shows the ground-truth image without noise. The iNBMT can be clearly seen to exhibit the best performance.


(a) Ground-truth without noise


(b) Corrupted ground-truth


(c) Reconstruction by iNBMT


(d) Reconstruction by ME-IBP


(e) Reconstruction by IBP-IBP

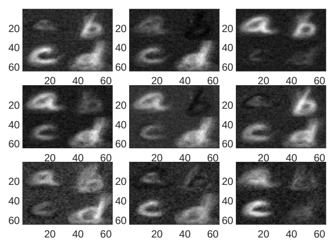

(f) Reconstruction by PNMT

Fig. 4.8: Comparison of iNBMT, ME-IBP, IBP-IBP, and PNMT on the Com-NIST dataset. The first sub-figure shows the ground-truth image without noise. The 3^{rd} to 6^{th} sub-figures show the reconstruction result. iNBMT clearly produces the best performance.

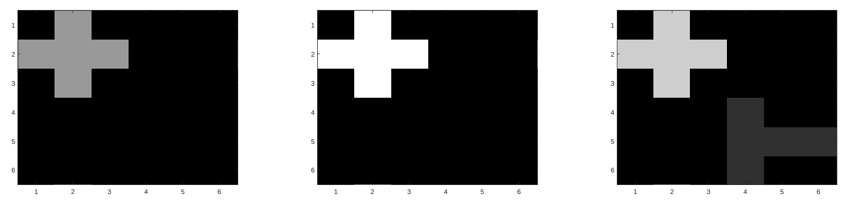
tween the reconstructed and ground-truth images without noise [153–156]. The von-Neumann divergence is defined as:

$$DvN(\mathbf{A}_1, \mathbf{A}_2) = tr(\mathbf{A}_1 \log \mathbf{A}_1 - \mathbf{A}_1 \log \mathbf{A}_2 - \mathbf{A}_1 + \mathbf{A}_2) \; .$$

Here, the $\mathbf{A}_1$ and $\mathbf{A}_2$ are two matrices. The von-Neumann divergence (NvD) has been shown to preserve the geometry information better when two images or matrices are compared. It is considered a closer measure to human visual perception [154, 155]. To verify this, we provide an illustrative example to explain the difference between the Mean Square Error (MSE), the Mean Absolute Error (MAE), and NvD.

Specifically, we use the MSE and MAE to measure the discrepancy of two images (as shown in Fig. 4.9). The image I can be seen to be more similar to image A than image B. However, $MSE(A, I) = 0.0222 > MSE(B, I) = 0.0111$ and $MAE(A, I) = 0.0556 > MAE(B, I) = 0.0550$, show that MSE and MAE may not be suitable. On the other hand, the von-Neumann divergence result $DvN(A, I) = 1.1325 < DvN(B, I) = 6.7802$ indicates that A is more similar to I, which is the same as the human visual perception.

In Table 4.2, we report von-Neumann divergence produced by the various methods. The proposed iNBMT model clearly leads to significantly smaller values than the other methods, showing that the iNBMT reconstructed images are more similar to the ground-truth. This result coincides with our intuition. as observed in Fig. 4.6, Fig. 4.7, and Fig. 4.8.



(a) Image A (b) Image I (c) Image B

Fig. 4.9: Illustration of von-Neumann divergence measure, which is more consistent with human visual perception than MAE and MSE.

4.5.4 Pre-image Restoration

In this section, we compare the performance of different algorithms for pre-image restoration. Latent features are first obtained for each model using the training set and the various features then evaluated in terms of their ability to restore test pre-images. The latter are intentionally halved.

Table 4.2: Reconstruction results by Von neumann divergence (the smaller the value, the better)

Datasets	Syn	Com-NIST	Com-USPS
iNBMT	**6.4530**	**310.2899**	**2598.5233**
ME-IBP	33.4758	1556.6001	5027.6282
IBP-IBP	60.4408	2416.1324	16893.2936
PNMT	216.1765	1958.5674	3737.1268

On the second row of Fig. 4.10 and Fig. 4.11, we again illustrate the ability of the four models to extract hidden features. Unlike previous experiments, here, each sample or an image consists of one single digit or letter rather than four combined digits or letters.

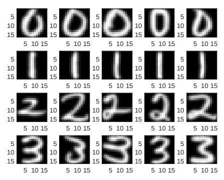
The various methods are first used to learn latent features, which are then exploited to restore incomplete images in the test set. Specifically, the features are learned from the training set, and binary matrix $\mathbf{Z}$'s updated with the test set. The new binary matrix $\mathbf{Z}$ contains $z_{nk} = 1$ if the n^{th} testing element is recognized as the k^{th} row feature. 20 incomplete digits are used in testing (Fig. 4.10(b)), and each row is the same number $(0-3)$. The recovered images are illustrated on the bottom row in Fig. 4.10. The sub-figures with red boxes are incorrectly restored. Similarly, for the Pre-NIST, each row of the test images denotes the same letter in Fig. 4.11(b). To evaluate the result, we only need to determine whether the number (letter) of each row is the label of the test image. From the results in both Pre-NIST and Pre-USPS, the iNBMT can be seen to have almost restored all the images correctly (except two errors in Pre-USPS and one error in Pre-NIST), while ME-IBP, IBP-IBP, and PNMT could not restore many of the images. This experiment demonstrates the advantages of our proposed iNBMT method.

Note that the above restorations were judged perceptually. Though subjective, it is sufficiently clear (to the naked eye) that images restored by our iNBMT model are much closer to the ground-truth images.

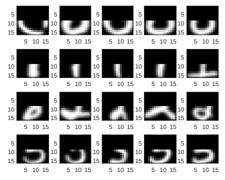
4.5.5 Clustering

In this section, we evaluate the clustering performance of the proposed iNBMT method on benchmark datasets, Coil-20 and Umist, compared with five related approaches, specifically, the classical clustering k-mean method, one-side clustering ME-IBP method, and state-of-the-art NMFT methods: PNMT and FNMTF [4]. The evaluation has been performed on the basis of three standard clustering criteria. c and c' have been set to true

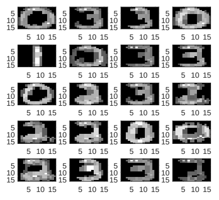
[4]Code can be downloaded from `https://github.com/lucasbrunialti/nmtf-coclustering`



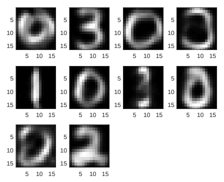
(a) 20 samples of training data



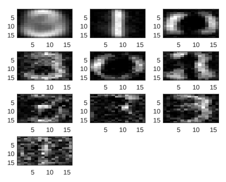
(b) Data for testing



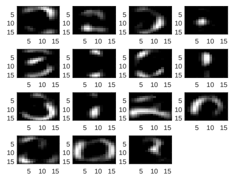
(c) Features learned by iNBMT



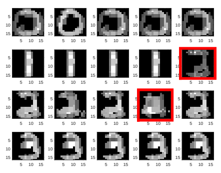
(d) Features learned by ME-IBP



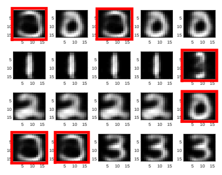
(e) Features learned by IBP-IBP



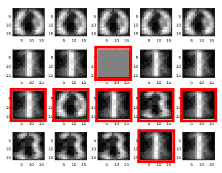
(f) Features learned by P-NMT



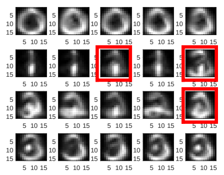
(g) Restored by iNBMT



(h) Restored by ME-IBP

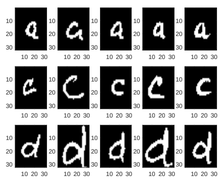


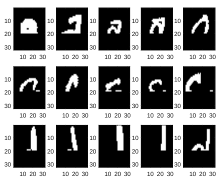
(i) Restored by IBP-IBP

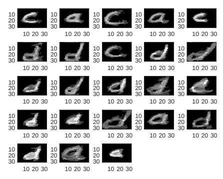


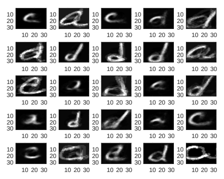
(j) Restored by PNMT

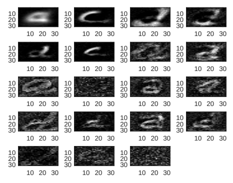
Fig. 4.10: Comparison of iNBMT, ME-IBP, IBP-IBP, and PNMT on the Pre-USPS dataset for pre-image restoration. The first row shows 20 samples of training data, and test images with five halved letters $(0, 1, 2, 3)$ are showing in each row respectively. The second row demonstrates the features learned by each method. The third row shows restoration results. Digits with red boxes are incorrectly restored.

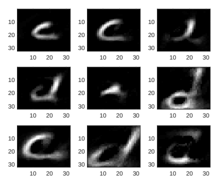

(a) 15 samples of training data

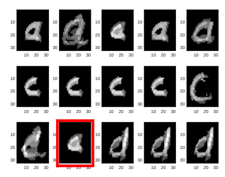

(b) Data for testing

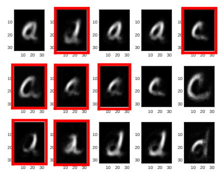

(c) Features learned by iNBMT

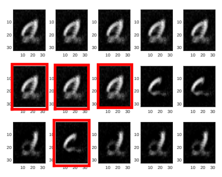

(d) Features learned by ME-IBP


(e) Features learned by IBP-IBP


(f) Features learned by PNMT


(g) Restored by iNBMT


(h) Restored by ME-IBP


(i) Restored by IBP-IBP

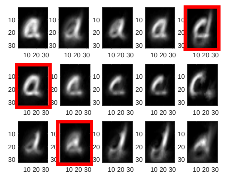

(j) Restored by PNMT

Fig. 4.11: Comparison of iNBMT, ME-IBP, IBP-IBP, and PNMT on the Pre-NIST dataset for pre-image restoration. The first row shows 15 samples of training data, and test images with five halved letters (a, c, d) are showing in each row respectively. The second row demonstrates the features learned by each method. The third row shows restoration results. Letters with red boxes are incorrectly restored.

labels and resulting cluster labels respectively, and N is the total number of samples. In the following, we describe the considered evaluation measures:

$$Accuracy = \frac{\sum_{i=1}^{N} \delta(c_i, map(c_i'))}{N} ;$$

where c and c' have been set to true labels and resulting cluster labels, N is the total number of samples, and $\delta(\cdot)$ denotes the Delta function, $\delta(x, y) = 1$ if $x = y$ and $\delta(x, y) = 0$ otherwise. We also map each cluster to an original label. This is used to measure the percentage of correctly clustered samples. The normalised mutual information (NMI) is used to measure the mutual information between two sets of clusters c and c'. It is also employed as an evaluation criterion.

$$NMI = \frac{I(c', c)}{(H[c] + H[c'])/2} , \ I(c', c) = \sum_{i \in c; j \in c'} p_{ij} \log_2 \frac{p_{ij}}{p_i p_j} .$$

Here, $H[c] = -\sum_{i=1}^{N} p_i \log_2 p_i$ and $p_{ij} = n_{ij}/N$ refers to the probability that a member in the cluster j belongs to class i, where n_{ij} is the number of members in cluster j belonging to class i. The purity measures percentage of total number of data points that were classified correctly:

$$Purity = \sum_{j \in c'} \frac{n_j}{N} p_j ,$$

where n_j is the number of all members in cluster j and $p_j = \frac{1}{n_j} max_i(n_{ij})$ [11].

The clustering experiment reports the results obtained by our method and four related approaches. The K-means and ME-IBP are one-sided clustering methods. The FNMTF and the PNMF are all co-clustering methods, but with the limitation that the number of clusters has to be specified. The FNMTF and PNMF set the number of clusters as the true number of classes and report the best average result. Results obtained in Tables 4.3 show that accuracy is significantly increased when co-clustering methods are applied. Based on the results of three criteria, the proposed method iNBMT can be seen to dramatically outperform the other benchmark methods.

4.6 Complexity Analysis

For measuring the inference efficiency, the time complexity per iteration will be calculated on a linear-Gaussian likelihood model. We show that the per-iteration complexity of our IBP based model outperforms other recently-proposed latent feature models [20][21].

In the ME framework, updating $p(\mathbf{Y}|\mathbf{Z})$ and $p(\mathbf{Y}|\mathbf{X})$ are independent of remaining observations and only require the computation of $\mathcal{T}(\cdot)$. Therefore, the $\mathcal{T}(\mathbf{Z})$ updating need $O(N(K^2 \ln K))$ operations, and $O(D(L^2 \ln L))$ operations are involved in

Table 4.3: Clustering results on two benchmark datasets (the higher, the better)

Data	Coil-20			Umist		
Metrics	Accuracy	NMI	Purity	Accuracy	NMI	Purity
iNBMT	**0.7911**	**0.8631**	**0.7917**	**0.5426**	**0.7028**	**0.5461**
k-means	0.5563	0.7768	0.6201	0.4696	0.5994	0.4713
ME-IBP	0.5792	0.6997	0.6042	0.4713	0.6128	0.4713
FNMTF	0.6937	0.7851	0.6944	0.4887	0.6355	0.4904
PNMT	0.6111	0.7554	0.6146	0.5009	0.6019	0.5026

updating $\mathcal{T}(\mathbf{X})$. In our proposed iNBMT model, it yields a per-iteration complexity of $O(N(K^2L) + D(L^2K))$ for updating $q(\mathbf{W})$, which consists of two parts: $O(K^2L)$ operations on optimising $p(\mathbf{Z})$, and $O(L^2K)$ operations on updating $p(\mathbf{X})$. Hence, the total per-iteration complexity of iNBMF yields $O(N(K^2L) + D(L^2K))$ operations. The latent feature model via IBP proposed in [20] uses similar ME inference over the latent factors. Its total per-iteration complexity of ME-IBP model is easily checked as $O(NK^2(D+\ln K))$ consisting of $O(K^2D)$ operations on $q(\mathbf{W})$ and $O(N(K^2\ln K))$ operations on the infinite variable $p(\mathbf{Y}|\mathbf{Z})$. Clearly, the operations of our model are mainly reduced when updating the parameters of non-negative matrices. In the correlated IBP-IBP model, there are also two infinite variables, the category $\mathbf{U}$ and latent features $\mathbf{V}$, the per-iteration complexity of $p(\mathbf{Y}|\mathbf{Z})$ is $O(NK^2L\ln N)$. In addition, $O(NK^2L(LD))$ operations are needed when updating $p(\mathbf{W})$. Therefore its total per-iteration complexity is about $O(NK^2L(LD + \ln N))$.

In practice, N and D are usually sufficiently larger than K and L.[5] Hence, the per-iteration complexity of the proposed iNBMT can be written in a simple form: $O(\alpha N + \beta D)$, while that of ME-IBP model is simplified to $O(\gamma ND)$ and IBP-IBP is simplified to $O(\delta ND)$, where α, β, γ, and δ are small coefficients. Clearly, our proposed iNBMT has the per-iteration complexity one order lower than that of the competitive models. For better comparison with the other feature-based models, we list the per-iteration complexity in Table 4.4.

4.7 Summary

In this chapter, we propose a new Bayesian model, termed infinite Non-negative Binary Matrix Tri-factorisation (iNBMT). The proposed model is proven to be capable of

[5]Observations are consist with N objects with D attributes.

Table 4.4: Number of parameters comparison. The per-iteration time complexity of our proposed iNBMT is one order lower than the other two models.

Methods	Per-iteration complexity
iNBMT	$O(\alpha N + \beta D)$
ME-IBP	$O(\gamma N D)$
IBP-IBP	$O(\delta N D)$

automatically learning latent binary features along with feature numbers, based on the Indian Buffet Process (IBP). The proposed iNBMT engages a tri-factorisation process that decomposes a nonnegative matrix into the product of three components, including two binary matrices and a non-negative real matrix. We impose an IBP prior on the two infinite binary matrices, while a truncated Gaussian distribution is assumed on the weight matrix. To optimise the model, we develop an efficient modified, variational Bayesian algorithm, with the iteration complexity one order lower than recently-proposed IBP-based models. We have carried out a series of experiments which demonstrate that our proposed iNBMT model significantly outperforms state-of-the-art algorithms on both benchmark and real-world data.

4.7.1 Limitation and Future Work

Despite the remarkable performance of our model, some limitations still need to be addressed. First, while our IBP based method is faster than most other approaches, it is not as fast as the traditional NMF. Second, as observed in experiments, repetitive binary features can sometimes be extracted. Both these issues will be addressed in future work to improve our iNMBT model further. Finally, similar to many factorisation methods, the proposed model can be applied to a range of future applications, including gene expression clustering [157, 158], graph matching [159], and zero-shot learning [160].

Chapter 5

A Novel Deep Density Model for Unsupervised Learning

Density models have been receiving much interest in multi-layered or deep architectures. In the models with a single hidden layer (or shallow models), with the increase of the component numbers, the scale of parameters will be significantly increased. When a model has massive parameters, it is difficult to find the optimal solution by using a gradient descent method. In the deep model, the layer-wise optimisation algorithm can be used to optimise the parameters in a layer-by-layer procedure, and then to obtain the optimal solution by fine-tuning. [29]. Compared to shallow models of a similar scale, deep models are able to reduce greatly the computational difficulties of learning, tend to resist overfitting through parameter sharing between successive layers [75], and can promote effectively cognitive capabilities [35, 78]. The knowledge of cognitive agents can be modelled using more complex probability distributions and a deep density model can provide better support for simulating complex data. Deep density models have theoretical and practical significance in many disciplines and have attracted considerable attention in prediction, reconstruction, clustering, and simulation [161]. Furthermore, probabilistic graphical models have always had a fundamental role in constructing or estimating sophisticated density in deep density models; these models include the restricted Boltzmann machine (RBM), Gaussian restricted Boltzmann machine (GRBM), and directed belief networks (DBNs) [12, 13, 162]. Despite their lower optimisation difficulties than shallow architectures, deep models still present computational difficulties in practice; for instance, RBMs are tricky to train with a large number of free parameters, while DBNs require costly inference procedures [14, 75, 163].

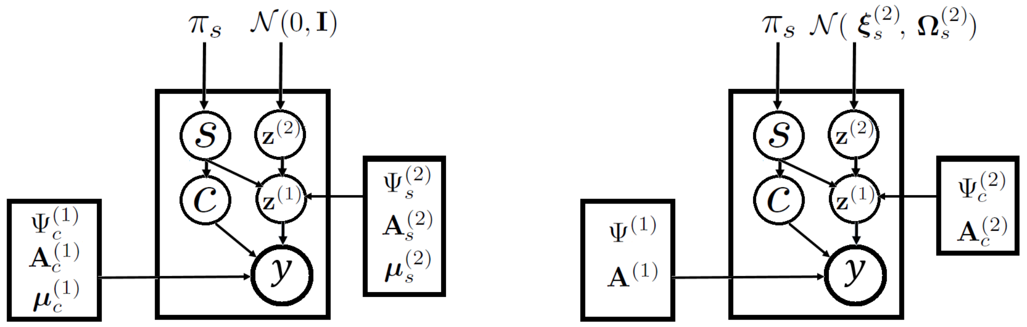
In this chapter, we look at two shortcomings of deep generative models: a large number of free parameters and costly inference procedures. Then the layer-wise learning approach is considered to address these concerns. This model is a directed graphical model which has been developed by adopting a shallow latent variable model. In particular, the deep architecture extends the same scheme in training each hidden layer. In addition,it takes the expectation-maximisation (EM) algorithm to maximise the log-likelihood in learning. Its inference and parameter computation procedure is more straightforward than previous methods.

The remainder of this chapter is structured as follows. The proposed method and inference procedure are described in Section 5.1. In Section 5.2, related work is briefly reviewed. In particular, the difference between our proposed method and other relevant work is emphasised. Section 5.3 demonstrates the density estimation and clustering results on four datasets and also includes the generation results on a benchmark dataset. The results obtained illustrate the improved performance of the proposed model (called DMCFA) over the other competitive models, e.g., MFA, DMFA, MCFA, and the shallow forms collapsed from the deep models. Finally, a conclusion is given in the last section. The content of this chapter has already been published on [22, 23].

5.1 Deep Mixtures of Factor Analysers with Common Loadings

This work addresses the previous limitations by proposing a novel greedy layer-wise learning approach, referred to as the Deep Mixtures of Factor Analysers with Common loadings (DMCFA). In developing the underlying idea, we extend the MFA model sharing a common component factor loading (MCFA) [100] when constructing a deep generative framework. The principal improvement is the common component factor loading which can be considered to be a feature selection or dimensionality reduction matrix. This, consequently reduces considerably the number of model parameters [164, 165]. DMCFA can simultaneously perform deep learning or clustering, together with dimensionality reduction or feature selection. In this case, a common loading can be well justified and is physically more meaningful. This setting can potentially further increase the performance, particularly in cases of large number of components or features [115]. The proposed model is also flexible in estimating the data density by utilising the learnable Gaussian distributions as the priors for each latent unit.

Fig. 5.1(b) offers an overview of the proposed model through a graphical example of a two-layer DMCFA. The first layer is constructed with two global parameters, which are the factor loading and the noise covariances. In the second layer, the common parameters are extended to each latent unit. While we introduce the mean and variance matrices of the latent factors, the number of free parameters is still dramatically reduced compared to that of deep mixtures of factor analysers (DMFA) [75]. Under a two-layer model, the total number of free parameters in DMCFA is far smaller than in DMFA; approximately, we



(a) Graphical model of DMFA.

(b) Graphical model of DMCFA.

Fig. 5.1: Graphical models of a two-layer DMFA and DMCFA. DMCFA is a deep directed graphical model utilising the multi-layer factor analysers which are developed by adopting an MCFA model in each hidden layer.

have $pq + cqd << cpq + sqd$, where $d < q \ll p$ (the dimensionality), and the total number of second-layer components s is a multiple of the total number of first layer components c. It is easy to verify that DMCFA utilises approximately only $1/c$ parameters of DMFA (details can be found in Section 5.1.4). Therefore, our model has the notable advantage of dealing with multivariate data with a larger number of clusters or with insufficient instances.

To optimise the proposed model, we further develop a simple yet efficient EM-type algorithm for both learning and inference. The modified EM algorithm converts the M-step into 2-layer loops and then performs learning in each layer independently. Compared to the classical MCFA model, the layers are configured to contribute to a simpler objective function within the EM algorithm with the same scale of mixtures [77]. This makes DMCFA inference and learning more straightforward and efficient and less likely to get trapped into a local optima. Meanwhile, the overfitting risk and the free parameters are reduced by sharing the factor loadings amongst the layers. With mild variational inference assumptions, when the bound is tight, any increase in the bound will improve the true log-likelihood of the model. Therefore, the higher layer has the ability to model a better aggregated posterior of the first layer, showing that the proposed deep model would be much better than training a shallow model.

5.1.1 Main Model

The MCFA model considered here is fundamental to the proposed method (detailed formulation in Chapter 3.1.2). We assume that a p-dimensional vector of observed variables $\mathbf{y}$ can be generated through a linear combination with a q-dimensional vector of latent factors $\mathbf{z}$ potentially corrupted by additive uncorrelated Gaussian noise $\boldsymbol{\epsilon}$. By introducing the common factor loadings $\mathbf{A} \in \mathbb{R}^{p\times q}$, the directed generative MCFA model is defined as

$$\mathbf{y} = \mathbf{A}\sum_{c=1}^{C}\mathbf{z}_c + \boldsymbol{\epsilon}, \quad \text{with probability } \pi_c \ (c = 1, \dots, C)\ . \tag{5.1}$$

The marginal density of the first-layer MCFA ($P_{\mathcal{MCFA}}(\mathbf{y};\boldsymbol{\theta})$) is given by a mixture of Gaussians with constrained mean and covariance, as

$$P_{\mathcal{MCFA}}(\mathbf{y};\boldsymbol{\theta}) = \sum_{c=1}^{C}\pi_c\mathcal{N}\left(\mathbf{y};\mathbf{A}\boldsymbol{\xi}_c, \mathbf{A}\boldsymbol{\Omega}_c\mathbf{A}^T + \boldsymbol{\Psi}\right)\ . \tag{5.2}$$

Here, the model parameters are denoted by $\boldsymbol{\theta} = \{\pi_c, \mathbf{A}, \boldsymbol{\Psi}, \boldsymbol{\xi}_c, \boldsymbol{\Omega}_c\}$, for $c = 1, \dots, C$.

The same scheme was extended to train the DMCFA model. By illustrating a mixture of the higher layer in Fig. 5.2, we can clearly see that a mixture model can better model a non-Gaussian posterior component in the lower layer. The key improvement is that the

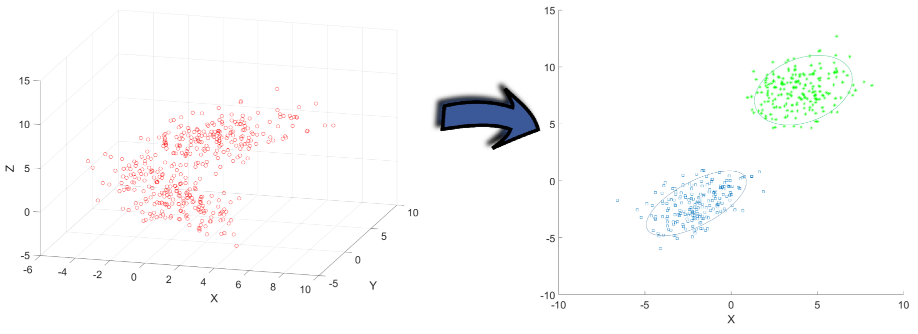


Fig. 5.2: The sketch of a mixture of the higher layer of DMCFA. **Left**: The aggregated posterior of a component in the lower layer is not a Gaussian distribution. **Right**: The higher layer has an ability to model a better aggregated posterior of the lower layer.

same load matrix is used for each mixture component that uses MCFA as an aggregated prior. This would potentially further improve the performance. Since different loading matrices are used in DMFA, the number of parameters may be unmanageable when the data has a larger feature dimensional and/or smaller observations. Moreover, different load matrices may not be physically essential or even unnecessary [23].

This section describes how to generalise an MCFA to multiple layers. In the shallow form, the latent factors in each component would be distributed according to a single Gaussian. However, this cannot describe the latent factors in practice, and the model can be improved by using a more powerful mixture of Gaussian priors. With this in mind, it is straightforward to form the second layer. Considering the same assumption as in MCFA, observed variables $\mathbf{y} \in \mathbb{R}^p$, latent factors $\mathbf{z} \in \mathbb{R}^q$, and non-overlapping components c are set in the first layer of the DMCFA model. After a shallow MCFA training reaches convergence, the prior of latent factors in the first layer will be replaced by an MCFA prior

$$p(\mathbf{z}|c) = P_{\mathcal{MCFA}}(\mathbf{z}_c^{(1)}; \boldsymbol{\theta}_c^{(2)}) \; . \tag{5.3}$$

Here, $\boldsymbol{\theta}_c^{(2)}$ denotes the new parameters for the second-layer MCFAs specific to the component c of the first layer. The same scheme can be extended to train more layers. In the second layer of the c^{th} component, $\mathbf{z}_c^{(1)}$ is the q-dimensional input pattern for the second layer, which is sample drawn using the posterior distribution of the first-layer latent factors

$$\begin{aligned} p(\mathbf{z}|\mathbf{y}, c) &= \mathcal{N}(\mathbf{z}; \boldsymbol{\kappa}_c, \mathbf{V}_c^{-1}) \; , \\ \mathbf{V}_c^{-1} &= \boldsymbol{\Omega}_c^{-1} + \mathbf{A}^T \boldsymbol{\Psi}^{-1} \mathbf{A} \; , \\ \boldsymbol{\kappa}_c &= \boldsymbol{\xi}_c + \mathbf{V}_c^{-1} \mathbf{A}^T \boldsymbol{\Psi}^{-1} (\mathbf{y} - \mathbf{A}\boldsymbol{\xi}_c) \; . \end{aligned} \tag{5.4}$$

In the deep model, $m_c \in \{1, \ldots, M_c\}$ is set to be the sub-component indicator variable[1], which is associated with the first-layer component c. In the second layer, the mixing proportion $\pi_{m_c}^{(2)}$ of component m_c is defined as

$$p(m_c) = \pi_{m_c}^{(2)} \; , \qquad \sum_{m_c=1}^{M_c} \pi_{m_c}^{(2)} = 1 \; . \tag{5.5}$$

The old MCFA prior is replaced by a new prior of DMCFA

$$p(\mathbf{z}, c) = p(c) p(\mathbf{z}|c) \Leftarrow p(\mathbf{z}, c) = p(c) p(m_c|c) p(\mathbf{z}|m_c) \; . \tag{5.6}$$

Different from the first layer where m_c is specific to the first-layer component, a simpler DMCFA formulation is established by enumerating all the second-layer components. A new indicator $s \in \{1, \ldots, S\}$ is denoted as the second-layer component indicator variable. S is the total number of the second-layer components satisfying $S = \sum_{c=1}^{C} m_c$. Therefore the new mixing proportions are given by

$$\pi_s^{(2)} = p(s) = p(c_s) p(s|c_s) \; , \qquad \sum_{s=1}^{S} \pi_s^{(2)} = 1 \; , \tag{5.7}$$

[1]One component of the first layer can be divided into M_c sub-components. The size of the sub-components in each first-layer component need not be the same.

where c_s is the first component associated with s, and every s belongs to one and only one c.

Specifically, the density on factors $\mathbf{z}^{(1)}$ follows the joint density over $\mathbf{z}^{(2)}$ and s according to

$$p(\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, s) = p(\mathbf{z}^{(1)}, c|\mathbf{z}^{(2)}, s)p(\mathbf{z}^{(2)}|s)p(s) \ . \tag{5.8}$$

It is also necessary to consider the following density functions

$$p(\mathbf{z}^{(1)}, c|s, \mathbf{z}^{(2)}) = \mathcal{N}\big(\mathbf{z}^{(1)}; \mathbf{A}_c^{(2)}\mathbf{z}^{(2)}, \mathbf{\Psi}_c^{(2)}\big) \ , \tag{5.9}$$

$$p(\mathbf{z}^{(2)}|s) = \mathcal{N}\big(\mathbf{z}^{(2)}; \boldsymbol{\xi}_s^{(2)}, \mathbf{\Omega}_s^{(2)}\big) \ . \tag{5.10}$$

The marginal distribution can also be written through a shallow form by integrating out the latent factors

$$p(\mathbf{y}|c) = \textstyle\int_{\mathbf{z}} p(\mathbf{y}|c, \mathbf{z})p(\mathbf{z}|c)d\mathbf{z} = \mathcal{N}(\mathbf{y}; \boldsymbol{\mu}_c, \mathbf{\Sigma}_c) \ , \tag{5.11}$$

$$\boldsymbol{\mu}_c = \mathbf{A}\boldsymbol{\xi}_c \ , \quad \mathbf{\Sigma}_c = \mathbf{A}\mathbf{\Omega}_c\mathbf{A}^T + \mathbf{\Psi} \ .$$

According to this equation, the Gaussian density can be expressed on the observed data $\mathbf{y}$ given $\mathbf{z}^{(1)}$ and c, as

$$p(\mathbf{y}|c, \mathbf{z}^{(1)}) = \mathcal{N}\big(\mathbf{y}; \mathbf{A}^{(1)}\mathbf{z}^{(1)}, \mathbf{\Psi}^{(1)}\big) \ . \tag{5.12}$$

Here, $\mathbf{A}^{(1)} \in \mathbb{R}^{p\times q}$, $\mathbf{\Psi}^{(1)} \in \mathbb{R}^{p\times p}$, $\mathbf{z}^{(1)} \in \mathbb{R}^{q}$, $\mathbf{A}_c^{(2)} \in \mathbb{R}^{q\times d}$, $\mathbf{\Psi}_c^{(2)} \in \mathbb{R}^{q\times q}$, $\mathbf{z}^{(2)} \in \mathbb{R}^{d}$, $\boldsymbol{\xi}_s^{(2)} \in \mathbb{R}^{d}$ and $\mathbf{\Omega}_s^{(2)} \in \mathbb{R}^{d\times d}$.[2] Conventionally, the joint distribution with the second layer-latent variables is given by

$$p(\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, s) = p(\mathbf{z}^{(1)}, c|\mathbf{z}^{(2)}, s)p(\mathbf{z}^{(2)}|s)p(s). \tag{5.13}$$

The same scheme can be extended to train the following layers of MCFA.

5.1.2 Inference

According to Eq. (5.4), we sample $\mathbf{z}^{(1)} \sim p(\mathbf{z}|\mathbf{y}, c)$ by selecting the component $\hat{c} = \underset{c}{\operatorname{argmax}} \ q(c|\mathbf{y}; \boldsymbol{\theta}_c)$. Then $\mathbf{z}^{(1)}$ and $\hat{c}$ are treated as the input data for the second layer. The posterior probability $q(s|\mathbf{z}^{(1)}, \hat{c}; \boldsymbol{\theta}_s)$ and the posterior distribution $p(\mathbf{z}^{(2)}, s|\mathbf{z}^{(1)}, \hat{c})$ are

[2]The superscript represents which layer these variables belongs to. Since in the second layer, the sub-components corresponding to a component of the first layer share a common loading and the variance of the independent noise, $\mathbf{A}_c^{(2)}$ and $\mathbf{\Psi}_c^{(2)}$ are marked with the subscript c. d denotes the d-dimensional subspace of the second layer, where $d < q$.

computed similarly by the inference for the Maximum A Posteriori (MAP). The formulations of the posterior distribution are a similar fashion in Chapter 2.2.5 & 2.2.6 with respect to the second-layer parameters[3]

$$q(\mathbf{z}^{(2)}, s|\mathbf{z}^{(1)}, c) = \mathcal{N}\big(\mathbf{z}^{(2)}; \boldsymbol{\kappa}_{c_s}^{(2)}, \mathbf{V}_{c_s}^{(2)^{-1}}\big) \;, \tag{5.14}$$

where

$$\begin{aligned} \mathbf{V}_{c_s}^{(2)^{-1}} &= \boldsymbol{\Omega}_s^{(2)^{-1}} + \mathbf{A}_c^{(2)^T} \boldsymbol{\Psi}_c^{(2)^{-1}} \mathbf{A}_c^{(2)} \;, \\ \boldsymbol{\kappa}_{c_s}^{(2)} &= \boldsymbol{\xi}_c^{(2)} + \mathbf{V}_{c_s}^{(2)^{-1}} \mathbf{A}_c^{(2)^T} \boldsymbol{\Psi}_c^{(2)^{-1}} (\mathbf{z}^{(1)} - \mathbf{A}_c^{(2)} \boldsymbol{\xi}_c^{(2)}) \;. \end{aligned}$$

The posterior of the components can be found as follows

$$\begin{aligned} q(s|\mathbf{z}^{(1)}, c) &\propto p(\mathbf{z}^{(1)}, c|s) p(s) \;, \\ \hat{\boldsymbol{s}} &= \arg\max_s p(s) q(s|\mathbf{z}^{(1)}) \;. \end{aligned} \tag{5.15}$$

In maximum likelihood estimation, the likelihood of the mixture model corresponding to the c^{th} component derived from the first layer is estimated concerning the new observations $\mathbf{z}_c^{(1)}$ and parameters $\boldsymbol{\theta}_{c_s}^{(2)}$

$$P(\mathbf{z}_c^{(1)}; \boldsymbol{\theta}_{c_s}^{(2)}) = \sum_{s \in c} \pi_s \prod_{i=1}^{q} \big\{ \mathcal{N}(\mathbf{z}_i^{(2)} | \mathbf{A}_c^{(2)} \boldsymbol{\xi}_c^{(2)}, \mathbf{A}_c^{(2)} \boldsymbol{\Omega}_s^{(2)} \mathbf{A}_c^{(2)^T} + \boldsymbol{\Psi}_c^{(2)}) \big\} \;, \tag{5.16}$$

where s is just allowed to belong to one and only one c, and q denotes the number of dimensions of the new observations.

The algorithm we proposed for training DMCFA is based on the EM algorithm. Since the mixtures are independent in a layer due to the greedy layer-wise optimisation, the EM algorithm can be used to estimate the parameters of each mixture and find a local maximum of the log-likelihood. Given the complete data $\mathbf{y}$, the first-layer log-likelihood objective function is formulated as

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}_c | \mathbf{y}, \omega_c, \mathbf{z}_c) = & \sum_{c=1}^{C} \int_{\mathbf{z}} q(\mathbf{z}, c | \mathbf{y}; \boldsymbol{\theta}_c) \\ & \big\{ \log p(\mathbf{y}|c, \mathbf{z}) + \log p(\mathbf{z}|c) + \log \pi_c \big\} d\mathbf{z} \;, \end{aligned} \tag{5.17}$$

where $q(\mathbf{z}, c|\mathbf{y}; \boldsymbol{\theta}_c)$ is the posterior distribution. The bound is tight when $q(\mathbf{z}, c|\mathbf{y}; \boldsymbol{\theta}_c) = p(\mathbf{z}, c|\mathbf{y}; \boldsymbol{\theta}_c)$. With regard to the second-layer parameters $\boldsymbol{\theta}_c^{(2)}$ which are specific to component c of the first layer, the DMFA formulation seeks to substitute a more effective prior $\log p(\mathbf{z}^{(1)}|\hat{c}; \boldsymbol{\theta}_c^{(2)})$. Holding the first-layer parameters fixed, maximising Eq. (5.17)

[3]The subscript emphasises the sub-component s which is specific to a component c of the first layer.

with the second-layer parameters is equivalent to maximising

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}_c^{(2)}|\mathbf{z}^{(1)}) =& \sum_{s=1}^{S}\int_{\mathbf{z}}^{(2)} q(\mathbf{z}^{(2)}, s|\mathbf{z}^{(1)}, \hat{c}; \boldsymbol{\theta}_c^{(2)})\big\{\log p(\mathbf{z}^{(1)}, \hat{c}|\mathbf{z}^{(2)}, s; \boldsymbol{\theta}_c^{(2)}) \\ &+ \log p(\mathbf{z}^{(2)}|s) + \log p(s)\big\} d\mathbf{z}\ .\end{aligned} \tag{5.18}$$

The second-layer parameter vector consists of $\boldsymbol{\theta}_c^{(2)} = \{\pi_s, \mathbf{A}_c, \boldsymbol{\Psi}_c, \boldsymbol{\xi}_s, \boldsymbol{\Omega}_s\}$, and $c = 1, \dots, C$. Parameters of the mixture model on the new observations $\mathbf{z}_c^{(1)}$ can be updated

$$\hat{\boldsymbol{\theta}}_{\boldsymbol{c}}^{\mathbf{(2)}} = \arg\max_{\theta_c^{(2)}} \mathcal{L}(\boldsymbol{\theta}_c^{(2)}|\mathbf{z}^{(1)}). \tag{5.19}$$

In the layer-wise learning scenario, an MCFA is trained in a standard way for the first layer. For the second layer, the parameters of the first layer become fixed, and new training data are sampled depending on the posteriors from the first layer. The modified EM-algorithm is developed to alter the M-step into 2-layer loops. The procedure is summarized in Algorithm 4.

Algorithm 4: The procedure of the 2-loop M-step

```
Input  : Initialized paremeters $\boldsymbol{\theta} = \{\pi_c, \mathbf{A}, \boldsymbol{\xi}, \boldsymbol{\Omega}_c, \mathbf{D}_c\}$, and the initial value of the
         log-likelihood.
Output: Optimal values of parameters $\boldsymbol{\theta}$.

M-step :
    Update the global parameters $\boldsymbol{\theta}_g = \{\mathbf{A}, \mathbf{D}\}$ :
        Re-estimate the parameters $\mathbf{A}, \mathbf{D}$ by maximisation of $\mathbb{E}[\mathcal{L}(\boldsymbol{\theta}_c|\mathbf{y}, \omega_c, \mathbf{z}_c)]$;

        Calculating partial derivatives of the expectation equations for each global
        parameters $\partial\mathbb{E}[\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(k)})]/\partial\boldsymbol{\theta}_g = 0$.

    Update the local parameters $\boldsymbol{\theta}_l = \{\pi_c, \boldsymbol{\xi}_c, \boldsymbol{\Omega}_c\}$ :
        for $c = 1$ to $C$ do
            Re-estimate the parameters $\boldsymbol{\theta}_l = \pi_c, \boldsymbol{\xi}_c, \boldsymbol{\Omega}_c$ by calculating partial
            derivatives of the expectation equations for each local parameters
            $\partial\mathbb{E}[\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(k)})]/\partial\boldsymbol{\theta}_l = 0$.
```

Note that, since $\mathbf{A}$ is orthogonal, any upper triangular matrix $\mathbf{U}$ can be absorbed in $\mathbf{A}$ by setting $\mathbf{A} \leftarrow \mathbf{A}\mathbf{U}^T$, where $\mathbf{U}$ is the Cholesky factor of $\boldsymbol{\Omega}_c$. Therefore, the updated estimates $\boldsymbol{\xi}_c$ and $\boldsymbol{\Omega}_c$ are adjusted by setting $\boldsymbol{\xi}_c \leftarrow \mathbf{U}\boldsymbol{\xi}_c$ and $\boldsymbol{\Omega}_c \leftarrow \mathbf{U}\boldsymbol{\Omega}_c\mathbf{U}^T$.

5.1.3 Collapse Model

Although a DMCFA model can be collapsed back into a standard shallow MCFA by multiplying the factor loading matrices at each layer, the learning of these two models is entirely different since the lower layer shares the parameters with the components of the upper layers in the deep model. Therefore, DMCFA is more efficient and straightforward than the shallow form, which attributes to the conditional distribution of the components in the previously hidden layer which is not modelled using the parameters of the following hidden layers. Moreover, the over-fitting risk and the computational cost of learning can be significantly reduced by sharing the factor loadings among the layers.

After the first-layer factors $\mathbf{z}^{(1)}$ are integrated out, we obtain a multivariate Gaussian density

$$p(\mathbf{y}|\mathbf{z}^{(2)}, s) = \mathcal{N}\big(\mathbf{y}; \mathbf{A}^{(1)}(\mathbf{A}_c^{(2)}\mathbf{z}^{(2)}), \mathbf{A}^{(1)}\boldsymbol{\Psi}_c^{(2)}\mathbf{A}^{(1)^T} + \boldsymbol{\Psi}^{(1)}\big) \ . \tag{5.20}$$

By further integrating out the second-layer factors $\mathbf{z}^{(2)}$, the final shallow form is then obtained by

$$\begin{gathered} p(\mathbf{y}|s) = \textstyle\int_{\mathbf{z}^{(2)}} p(\mathbf{y}|\mathbf{z}^{(2)}, s)p(\mathbf{z}^{(2)}|s)d\mathbf{z}^{(2)} = \mathcal{N}(\mathbf{y}; \mathbf{m}_s, \boldsymbol{\Sigma}_s) \ , \\ \mathbf{m}_s = \mathbf{A}^{(1)}(\mathbf{A}_c^{(2)}\xi_s^{(2)}) \ , \quad \boldsymbol{\Sigma}_s = \mathbf{A}^{(1)}(\mathbf{A}_c^{(2)}\boldsymbol{\Omega}_s^{(2)}\mathbf{A}_c^{(2)^T} + \boldsymbol{\Psi}_c^{(2)})\mathbf{A}^{(1)^T} + \boldsymbol{\Psi}^{(1)} \ . \end{gathered} \tag{5.21}$$

Finally, the marginal density of the shallow model is given by a mixture of Gaussians with the complete data $\mathbf{y}$, given as

$$p(\mathbf{y}) = \sum_{s=1}^{S} p(s)p(\mathbf{y}|s) = \sum_{s=1}^{S} \pi_s \mathcal{N}(\mathbf{y}; \mathbf{m}_s, \boldsymbol{\Sigma}_s) \ . \tag{5.22}$$

Conventionally, $\boldsymbol{\theta}_{c_s} = \{\pi_s, \mathbf{A}, \boldsymbol{\Psi}, \mathbf{A}_c, \boldsymbol{\Psi}_c, \boldsymbol{\xi}_s, \boldsymbol{\Omega}_s\}_{s=1,c=1}^{S,C}$ represents the parameters of this shallow MCFA. The posterior distribution of the latent factor $\mathbf{z}_s$ can also be collapsed to a shallow form

$$\begin{gathered} q(\mathbf{z}_s, s|\mathbf{y}) = \mathcal{N}(\mathbf{z}_s; \boldsymbol{\kappa}_s, \mathbf{V}_s^{-1}) \ , \\ \mathbf{V}_s^{-1} = (\mathbf{A}_c^{(2)}\boldsymbol{\Omega}_s\mathbf{A}_c^{(2)} + \boldsymbol{\Psi}_c^{(2)})^{-1} + \mathbf{A}^{(1)^T}\boldsymbol{\Psi}^{(1)^{-1}}\mathbf{A}^{(1)} \ , \\ \boldsymbol{\kappa}_s = \mathbf{A}^{(1)^T}\boldsymbol{\xi}_s + \mathbf{V}_s^{-1}\mathbf{A}^{(1)^T}\boldsymbol{\Psi}^{(1)^{-1}}(\mathbf{y} - \mathbf{m}_s) \ . \end{gathered} \tag{5.23}$$

5.1.4 Complexity Analysis

To measure the inference efficiency, the time complexity per iteration is calculated on the E-step and M-step. It is shown that the per-iteration complexity of our model outperforms the standard EM-algorithm and the free parameters are further reduced compared with the recently proposed deep mixture model.

The following calculations are all based on the two-layer mixture model, in which n and p denote the sample size and the number of dimensions, c and s are the numbers of mixtures in the first and second layer, and q and d are the dimensions of representation space in the first and second layer, respectively. A rough estimation of the computation complexity is calculated similar to [166]. With a multi-linear Gaussian likelihood model, both DMFA and DMCFA have same per-iteration complexity in E-step: $O(cn(p+q+1))$ on the first layer and $O(sn(q+d+1))$ on the second layer. DMFA yields $O(3cnp)$ and $O(3snq)$ operations on the first and second layers per-iteration by using the standard M-step. In our proposed DMCFA model, an efficient M-step is developed by updating the global parameters and local parameters alternatively. DMCFA yields $O(2np+2cnq)$ operations in the first step and $O(2nq+2snd)$ operations in the second step. Clearly, the operations of the proposed model are mainly reduced when updating the global parameters of the common factor loadings.

Since a common component loading matrix is shared across the components, the number of free parameters is dramatically reduced when compared with DMFA, even though the mean and variance matrices of latent factors are introduced. In practice, the diagonal matrix covariance just has p parameters, and the covariance matrix contains $\frac{q(q-1)}{2}$ constraints. From all the above settings, the total numbers of parameters are

$$T_{DMCFA} = s - 1 + p + pq - q^2 + c\left[2q + \frac{q(q+1)}{2} + qd - d^2\right] + s\frac{d^2+3d}{2} \; .$$

$$T_{DMFA} = s - 1 + c\left[2p + pq - \frac{q(q-1)}{2}\right] + s\left[2q + qd - \frac{d(d-1)}{2}\right] \; .$$

Practically, in the deep mixture models, it is usually the case that $p \gg q > d$, $c > 1$, and the second-layer component number s must be a positive integer multiple of c. T_{DMCFA} can be roughly given as $p(1+q)$, while T_{DMFA} is approximately $c(2+q)p$. Hence, $T_{DMCFA}/T_{DMFA} = (1+q)/(2c+qc) < 1/c$, which means that the proposed DMCFA merely uses $1/c$ parameters of DMFA.

5.2 Benchmarking Approaches

The work most related to the proposed model is the deep mixtures of factor analysers (DMFA), which is a deep directed graphical model utilising the multi-layer factor analysers developed by adopting an MFA in each hidden layer [31, 75]. MFA introduces a multivariate standard normal prior that is specified via the latent factors for all components. The principal method is to sample the data regarding the posterior distributions of the current layer and treat it as the training data for the next layer. Fig. 5.1(a) presents an instance of DMFA, where the observation vector and the first hidden layer are treated as an MFA component, while the parameters are used for the learning. After fixing the first-layer parameters, the priors of next-layer MFAs are replaced by sampling the hidden units of the current layer. The same scheme can be extended to the training of the subsequent layers. Importantly, different loading matrices are exploited for the different components in DMFA. Therefore, if the data has large feature dimensionality and/or small number of observations, the number of parameters may not be manageable. On the other hand, different loading matrices may be even less physically meaningful. In comparison, the proposed DMCFA model adopts the common component factor loadings to cope with these situations and provides both theoretical and empirical justification for its effectiveness.

The proposed model is also compared with several classical shallow models. With regard to the model itself, a deep MCFA structure can be collapsed into a standard shallow MCFA by multiplying the factor loading matrices at each layer. However, since the lower layer shares the parameters with the components at the upper layers, the learning of these two models is entirely different. For a shallow model, large-scale mixtures could render the objective function sufficiently complex. On the other hand, the model parameter redundancy may also lead to overfitting during learning. If we assume that $d < q \ll p$ denotes the number of the second and the first layer factors, and that the number of attributes and the second layer component number are an a multiple of the first-layer component number, then we have $s > ac$. A standard shallow MFA has s components and q factors. The reduced parameters are then calculated as $T = spq - (cpq + sqd) = q\,(p(s-c) - ds) > 0$.

5.3 Experimental Results

In this section, we evaluate the DMCFA's performance for model-based density estimation and clustering using five real datasets with: two standard models MFA and MCFA, a deep model DMFA (detailed shown in Chapter 2.4.2), and the shallow forms collapsed by the deep models (details in Chapter 2.4.4 & 5.1.3). Moreover, we conduct a qualitative experiment on a benchmark dataset to evaluate the performance in terms of generation.

5.3.1 Experimental Setup

In the experiment, we follow the work of DMFA [75], where according to their findings, adding a third layer can only bring little value. Therefore, we implement two layers for all deep models and for all experiments. The same mixture settings are used in the second layer. To reduce clutter, the scenarios of density estimation and clustering exploit the same parameter settings. The detailed settings of two deep models are described in

Table 5.1: Parameter settings. ♯Layers denotes the layer numbers of the deep models. MIX and FAC denote the number of mixture components and the factors, respectively.

Dataset	♯Layers	DMFA		DMCFA	
		MIX	FAC	MIX	FAC
ULC-3	1	3	90	3	90
	2	6	20	6	8
Coil-4-proc	1	4	16	4	16
	2	8	8	8	12
Leuk72_3k	1	3	16	3	16
	2	6	6	6	6
USPS1-4	1	4	16	4	10
	2	8	8	8	8

Table 5.1. These settings are achieved by using a trial and error approach to get the best empirical performance. For the standard MFA and MCFA, we set the same number of mixture components and factors, with the first layer of their "deep" counterparts.

To assess the model-based clustering performance, we compute the error rate (ERR) defined as

$$ERR = 1 - \frac{\sum_{i=1}^{N} \delta(c_i, map(c_i'))}{N} ,$$

where $\delta(x, y) = 1$ if $x = y$ and $\delta(x, y) = 0$ otherwise, and N denotes the total number of observations. The true labels and the result cluster labels are set to c and c', respectively. Smaller values indicate better clustering performance.

5.3.2 Datasets Description

We use an artificial dataset, a multivariate physical dataset and 3 image collection datasets to evaluate the proposed model throughout this section. The details are listed as below:

- ULC-3: The urban land cover (ULC) data consists of 3 types with 273 training samples, 77 test samples, and 147 attributes which are collected by classifying a high-resolution aerial image [106, 107].
- Leuk72_3k: This is an artificial dataset drawn from randomly generated Gaussian mixtures [167]. There are 3 classes with 39 attributes including 54 training samples and 18 test samples.
- Coil-4-proc: This dataset is a collection of 4 objects consisting of gray-scale images and each object has 72 samples [108]. These images discard the background

Table 5.2: Performance on various real data in terms of the average log-likelihood (the larger, the better) on the training set. DMFA and DMCFA are each set to two layers. S-MFA and S-MCFA denote the shallow form by collapsing the deep models.

Training Results						
Dataset	MFA	MCFA	DMFA	DMCFA	S-MFA	S-MCFA
ULC-3	-434.0364	-216.7516	-98.4912	**-89.2465**	-424.0828	-212.7775
Coil-4-proc	-3813.9950	-1521.7157	-24.3242	**-10.8630**	-3759.5424	-1494.3084
Leuk72_3k	-154.8025	-117.4220	-16.9911	**-12.8301**	-152.1288	-114.1186
USPS1-4	-1078.2457	-451.4169	**-14.9465**	-19.6103	-1073.4724	-442.0988

and downsampled to resolutions of 32×32. We reshaped each image to a 1024-dimensional vector. We divide the data into a training set and a test set, with 248 samples and 40 samples separately by randomly sampling.

- USPS1-4: This dataset contains 16×16 images of 1 to 4 handwriting digits of size pixels. Each image is reshaped to a vector, hence, the dimensionality is equal to 256. Both the training set and the testing set include 100 (random sampled) images of each digit.

- MNIST: The benchmark handwritten digits dataset which is composed of $60,000$ 28×28 handwriting digit images [168].[4]

5.3.3 Results

Empirical results

We first examine the quality of the density evaluation produced by the DMCFA and the other rival methods on a variety of datasets. Empirically, the log-likelihood value can reflect the fitting degree of the model parameters. Hence, the average log-likelihood is exploited to evaluate the quality of density estimation. Table 5.2 lists the mean log-probability values for the deep models versus the shallow models on their training set. The DMCFA has a better performance in training on most datasets except the USPS1-4 dataset. On the USPS1-4 dataset, DMFA has the highest value in training, but it cannot gain the best clustering result (in Fig. 5.5). This may be caused by overfitting due to the limited samples. The average log-probabilities values on the test set are listed in Table 5.3
.

[4]http://yann.lecun.com/exdb/mnist/

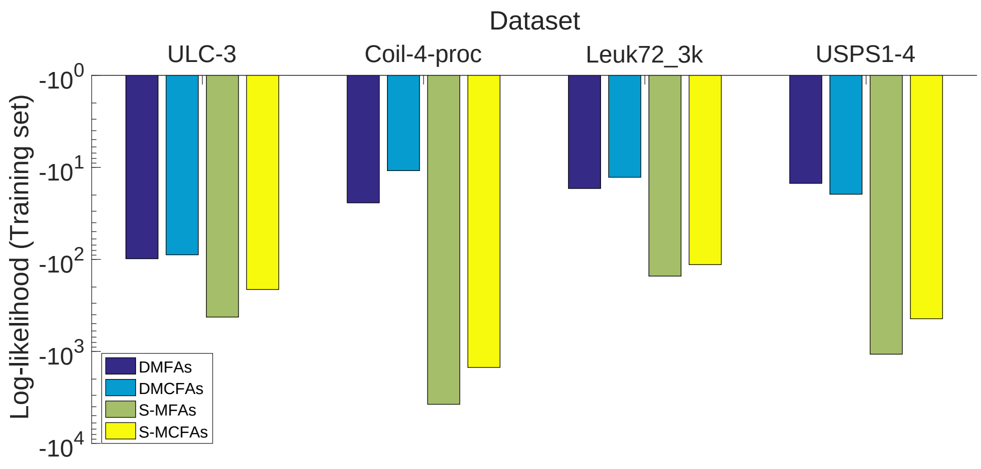


Fig. 5.3: Performance on various real data (on the training set) in terms of the average log-likelihood (the larger, the better). DMFA and DMCFA are all set in two layers. S-MFA and S-MCFA denote respectively the shallow form by collapsing the deep models.

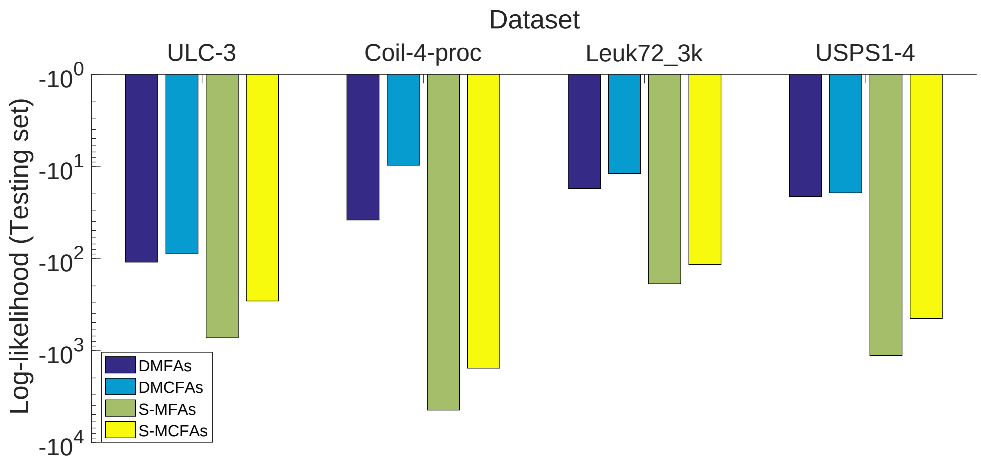


Fig. 5.4: Performance on various real data (on the testing set) in terms of the log-likelihood (the larger, the better). DMFA and DMCFA are all set in two layers. S-MFA and S-MCFA denote respectively the shallow form by collapsing the deep models.

Table 5.3: Performance on various real data in terms of the average log-likelihood (the larger, the better) on the test set. DMFA and DMCFA are each set to two layers. S-MFA and S-MCFA denote the shallow form by collapsing the deep models.

Test Results						
Dataset	MFA	MCFA	DMFA	DMCFA	S-MFA	S-MCFA
ULC-3	-737.4383	-295.4271	-110.1660	**-89.6897**	-732.2352	-291.4142
Coil-4-proc	-4504.3847	-1570.5914	-38.4105	**-9.7465**	-4466.2764	-1564.2895
Leuk72_3k	-191.3275	-120.6031	-17.4814	**-11.9715**	-189.5890	-117.2442
USPS1-4	-1140.2889	-461.5310	-21.2674	**-19.4758**	-1137.2793	-451.6439

In order to make the results of the deep model and the collapse model more visible, we conduct the empirical analysis through illustration. Here, all the results are all the results are logarithmically computed which makes the results of different datasets more conveniently compared. The average log-likelihood value on training data are shown in Fig. 5.3 by multiple trials. The results of the testing data are also obtained without updating parameters, as shown in Fig. 5.4. we can observe that the deep models can improve the true log-likelihood of the standard model dramatically.

Clearly, the DMCFA demonstrates better performance on all the datasets. By comparing the results of each group of training and testing, we can see that DMCFA is more capable in resisting overfitting. These results also reveal that the proposed deep model improves the true log-likelihood of the standard model dramatically.

Clustering results

To assess the model-based clustering performance, we compute the error rate on 4 real datasets for comparing the performance of DMCFA with the other methods. In the experiments, all of the methods have been initialised by random assortment. The clustering results for the deep models versus the shallow models on the training sets are shown in Fig. 5.5, and the results on the test sets are shown in Fig. 5.6. Also, we have done many trials to choose the best result by dropping attributes into different dimensions. Table 5.3 shows the detailed numerical results. The results shown here are the best from each model in the most appropriate dimensions. For each approach, the best results are reported, and it can be seen that the lowest error rate is obtained consistently with DMCFA which outperforms the other competitors. Although the DMFA and the shallow form collapsed by the DMCFA achieve the lowest error rate in some training sets, the results deteriorate

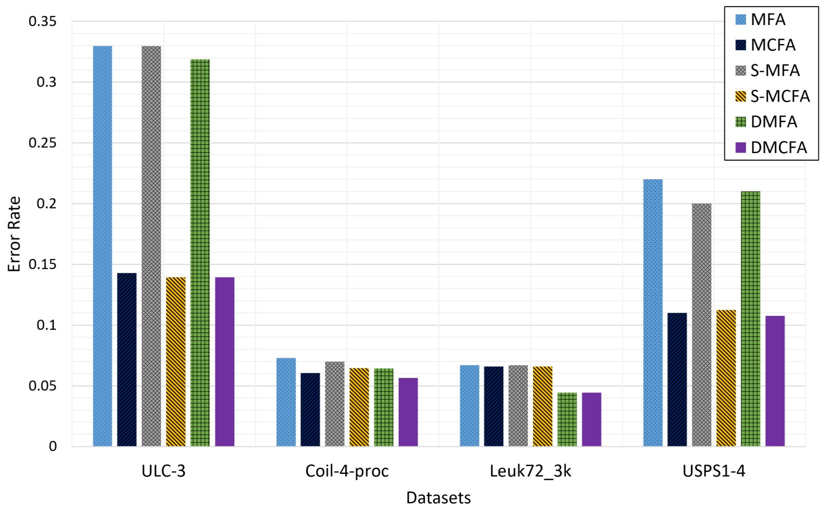


Fig. 5.5: Clustering performance (error rate) on all the datasets. The best result is reported from each model on the training set.

in the test set. Moreover, it can be clearly observed that the collapsed shallow form-s hardly improve performance. Also, the deep models are consistently better than their shallow counterparts. This observation further confirms the advantages of deep models over shallow ones.

Table 5.4: Clustering accuracy (error rate) on all the datasets. The best result is reported from each model on the training and testing sets.

Error Rate												
Dataset	MFA		MCFA		DMFA		DMCFA		Shallow MFA		Sallow MCFA	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
ULC	0.3297	0.2857	0.1429	0.1818	0.3187	0.2597	**0.1392**	**0.1558**	0.3297	0.2727	**0.1392**	0.1948
Coil-4-proc	0.0726	0.1250	0.0605	0.0500	0.0645	0.1000	**0.0565**	**0.0250**	0.0726	0.2000	0.0645	0.0500
Leuk72_3k	0.0667	0.1111	0.0667	0.0741	**0.0444**	0.0741	**0.0444**	**0.0370**	0.0667	0.1111	0.0667	0.0741
USPS1-4	0.2150	0.2375	0.1100	0.1725	0.2100	0.2325	**0.1075**	**0.1400**	0.2075	0.2350	0.1125	0.1475

Qualitative results

To demonstrate the generation results of all models, a qualitative study is conducted on the MNIST dataset. [169]. In this experiment, we try to generate the handwritten digits with a two-layer model given by DMCFA and DMFA. The deep models are trained with 10 first-layer components and 64 first-layer factors. Furthermore, we stack the second layer with 40 components (four components for each of the 10 first-layer components) and 16

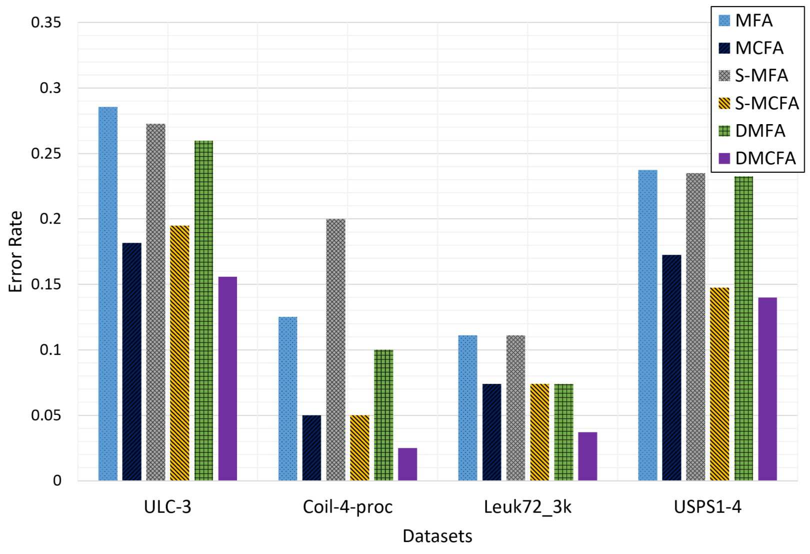
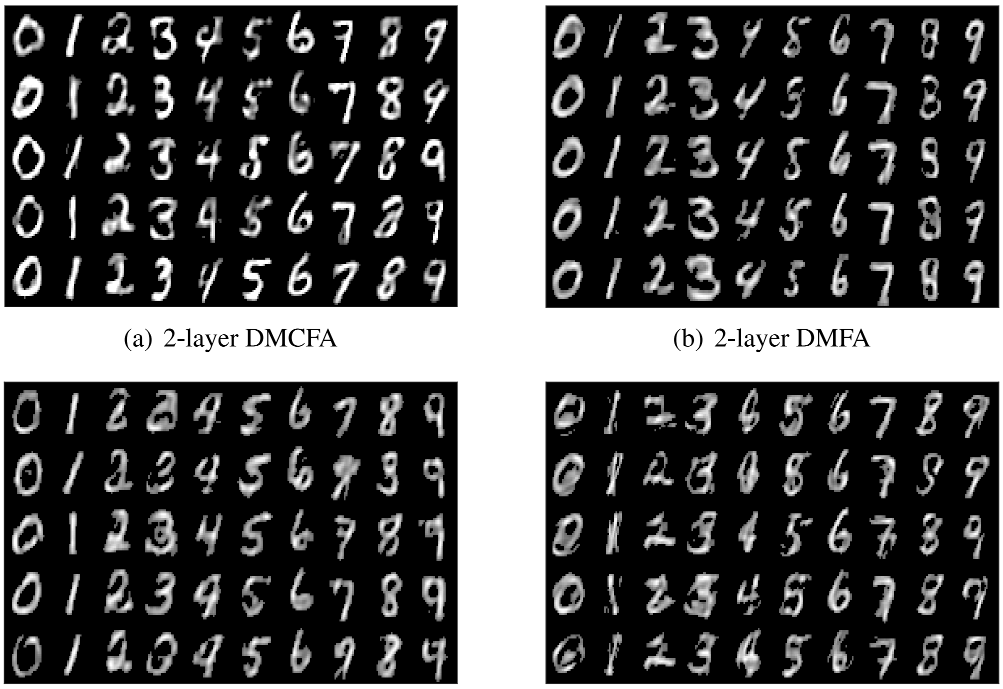


Fig. 5.6: Clustering performance (error rate) on all the datasets. The best result is reported from each model on test set.

factors. The process of generating a sample is as follows. First, we sample the variables within the latent space. Then, we can generate real samples from the density estimation in the observation space according to the reversibility property of the generative model. Fig. 5.7 presents some of the generated digits, with each column showing the same digit (since the mixture model can aggregate each type of numbers). From the results shown in Fig. 5.7(a), although the generated 4's and 9's could be easily confused, we can still clearly observe that the DMCFA model is able to generate a variety of samples with high quality. The results of the comparison model DMFA are shown in Fig. 5.7(b). The generated digits are similar, although they also have high quality. Here, we also compare the results of the shallow models, as shown in Fig. 5.7(c) and Fig. 5.7(d). Obviously, the results of the deep models are complete and more clear.

5.4 Summary

A novel deep density model, the DMCFA, is presented in this paper. Our approach borrows ideas from deep learning, multilayered factor analysis, and Gaussian mixture modeling to ensure that the learned density models for high-dimensional data are tractable. Exploiting the greedy layer-wise algorithm, we design an efficient expectation-maximisation algorithm to maximise the posterior and learn the parameters. Compared with existing deep density models, this approach enjoys an easy inference procedure, lower time complexity, and a significantly smaller number of free parameters. We evaluate our model on


(a) 2-layer DMCFA (b) 2-layer DMFA

(c) MCFA (d) MFA

Fig. 5.7: Comparison of two-layer DMCFA, two-layer DMFA, MCFA, and MFA on the MNIST dataset for generation.

empirical and clustering tasks using real-world datasets, which is shown to achieve better results compared to the state-of-the-art methods. Our generative model also allows us to produce real samples via sampling within the latent representation space of the learned clusters.

5.4.1 Limitation and Future Work

There are also some disadvantages of the proposed model. First, it may not be flexible enough, since it is required to pre-specify the parameters. Moreover, it is not natural to model arbitrarily non-linear representations with a mixture as used in the model. Second, the performance may be degraded when the involved layers are too many.

Future work includes the extension of the common loading idea to other deep density models. Also, an investigation of Bayesian methods will be conducted so that the number of mixtures and dimensions (of the latent representation space) can be automatically learned. Lastly, we intend to introduce the Neural Network framework so as to achieve an arbitrarily non-linear representation and deeper architecture (in Chapter 6).

Chapter 6

Deep Neural Network-Based Models via Density Estimation

Deep learning is perhaps the most popular learning method and a powerful tool for building intelligence systems. As deep neural networks, deep learning also can be considered as a nonlinear function but with a large number of parameters across various layers [170–172]. In contrast with shallow neural networks, there have been substantial improvements for deep models in the computational, algorithmic, and architectural aspects. Particularly, advances in software tools or platforms (e.g., Tensorflow and Torch) have enabled more convenience in building deep networks easily.

Compared with the traditional machine learning method, especially the probability models as the core of this dissertation, deep learning can realise multi-layer nonlinear mappings with a relatively mature and standard inference process. Moreover, deep learning can usually lead to excellent performance in many benchmark tasks. Despite these advantages, it is also widely recognised that there are still some limitations to the deep learning systems. These limitations or flaws are shown in Fig. 6.1 [173–175].

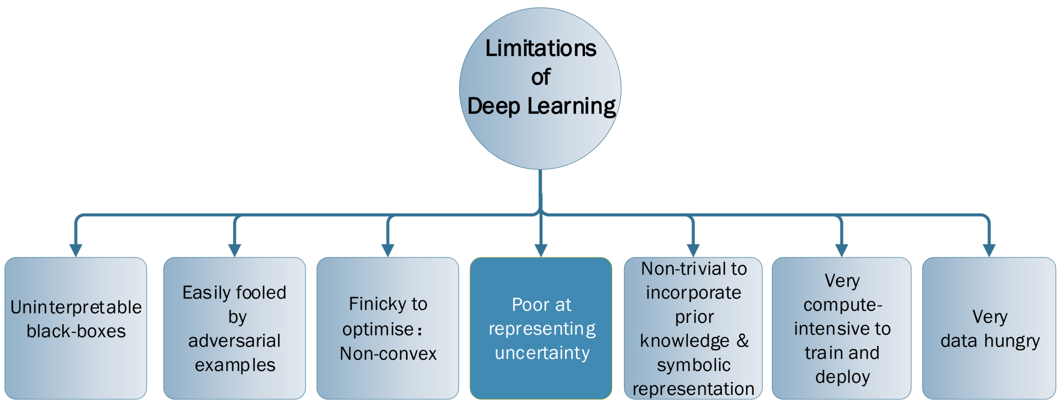

Limitations of Deep Learning
Uninterpretable black-boxes
Easily fooled by adversarial examples
Finicky to optimise: Non-convex
Poor at representing uncertainty
Non-trivial to incorporate prior knowledge & symbolic representation
Very compute-intensive to train and deploy
Very data hungry

Fig. 6.1: Flaws of the deep learning systems.

Among these shortcomings, how to represent uncertainty is the concern of our research. First of all, the learning models are expected to make predictions with an uncertainty measure. In another word, it is desirable if the models can know when they do not know, rather than giving a wrong answer directly (as the current deep models usually make a hard decision). To alleviate this problem, it is very important if the model can generate a probability while the decision is made [176]. On the other hand, probabilistic models are usually easy to manipulate in terms of complexity control or structure learning. Moreover, these models can even avoid the need for regularisation which is normally used to prevent overfitting.

In this chapter, for the purpose of enabling the advantages caused by the probabilistic approaches, we intend to develop several novel deep models by combining the density models with the deep Autoencoder. More specifically, in Section 6.1, a deep generative model is developed based on our previous work which extends the 2-layer joint learning model into a deep neural network framework. This model can be referred to as an Deep Autoencoder-Based Joint Learning Model. Section 6.1.1 and Section 6.1.2 will describe the model and the training strategy. Section 6.2 will then report the qualitative and quantitative results comparing our proposed new models against the other competitive methods on benchmark data. Also the network configuration will be presented in Section 6.2.2. Section 6.3 finally shows the discussion and future work.

6.1 A Deep Autoencoder-Based Joint Learning Model

In recent years, due to the great improvement in both hardware and the optimisation, deep learning has quickly become the most popular method in the learning community. Deep Autoencoder-Based Model is one of the generative frameworks which is an essential branch of deep learning.[1] Since Variational Autoencoder (VAE) is proposed to make density estimation on latent variables, the embedding of the density estimation into the autoencoder framework has been promoted as a hot topic. Many variants of VAE have been proposed to perform unsupervised clustering, semi-supervised classification, and supervised classification through deep generation models [80, 81, 177]. These recent advances show that these deep generative models and approximate Bayesian inference can significantly improve the flexibility, efficiency, and scalability of the traditional models. The autoencoder framework can be widely applied in classic applications such as the speech/image recognition, identifying and restoring damaged and contaminated images, and detection/rejection of anomalies.

In this section, a novel method is proposed with the goal of performing joint learning

[1]Another generative framework is based on the Generative Adversarial Networks

through deep generative models, which well combines Deep Autoencoder and density model. This approach, called the autoencoder-based joint learning model, can be considered as an enhanced version of our previous work 2L-MJFA as stated in Chapter 3. With the same motivation, the difference lies at the point that the traditional autoencoder (AE) is developed as a dimension reduction (DR) method. Similarly, though the independent realisation of AE and GMM can be easily implemented, it may notably diminish the final performance as the optimal low-dimensional subspace obtained by the AE may not be maximally beneficial to the targeted density estimation task [8, 9]. In comparison, the deep autoencoder-based joint learning network interacts the AE with GMM together which would optimise the parameters of both the models simultaneously. Such a joint optimisation can well balance autoencoding reconstruction, latent representation of density estimation, and regularisation terms. This helps the autoencoder to escape the less attractive local optimal solution and further reduces the reconstruction error. Importantly, this network is designed in an end-to-end fashion that does not require pre-training the autoencoder network [83, 85]. Another merit with the proposed model is that the standard backpropagation can be straightforwardly applied due to convex nature of the energy function (a part of the loss function), enabling an easy yet effective model inference process.

6.1.1 Model Description

The proposed Deep Autoencoder-Based Joint Learning Model is an end-to-end approach for supervised recognition and rejection. It utilises a deep autoencoder to generate a low-dimensional representation, which is further used to perform density estimation. The model structure is shown in Fig. 6.2, which consists of two networks.

The DR network performs low-dimensional representation $\mathbf{z}$ for input image $\mathbf{x}$ by a deep convolutional autoencoder

$$\mathbf{z} = h(\mathbf{x}; \boldsymbol{\theta}_e) \ , \qquad \mathbf{x}' = g(\mathbf{z}; \boldsymbol{\theta}_d) \ , \tag{6.1}$$

where an input image is encoded by the encoding function $h(\cdot)$ with parameter $\boldsymbol{\theta}_e$, and an image $\mathbf{x}'$ is reconstructed by the decoding function $g(\cdot)$ with parameter $\boldsymbol{\theta}_d$. The loss function is given by an L_2-norm that characterises the reconstruction error caused by the deep autoencoder

$$L_{AE}(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_2^2 \ . \tag{6.2}$$

The classification network performs c-dimensional probability vector for input low-dimensional representations $\mathbf{z}$ by a multi-layer fully-connected network

$$p = MLP(\mathbf{z}; \boldsymbol{\theta}_p) \ , \qquad p(c) = softmax(p) \ . \tag{6.3}$$

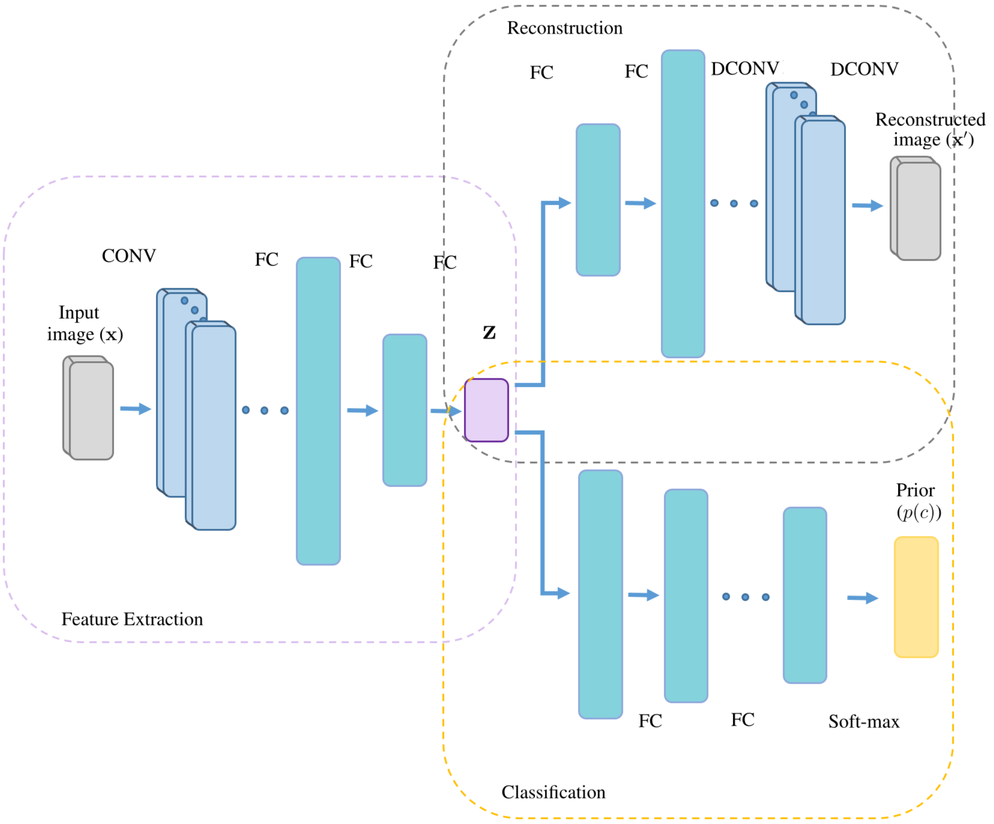


Fig. 6.2: Schematic structure of a Deep Autoencoder-Based Joint Learning Model. Conventionally, COVN denotes the convolutional layer, DCOVN denotes the deconvolutional layer, and FC denotes the fully-connected layer.

$p(c)$ is denoted as the class prediction of each image and is exploited as the only latent variable for the subsequent Gaussian mixture model, where p is the output of the multi-layered perceptions $MLP(\cdot)$, $softmax(\cdot)$ denotes the soft-max function. By considering a mixture of C multinomial distributions, the density of $\mathbf{z}$ could be modelled as a finite mixture model. The mixing proportion $\boldsymbol{\phi}_c$, mean $\boldsymbol{\mu}_c$, and covariance $\boldsymbol{\Sigma}_c$ for the c^{th} class in GMM are calculated by the following equations

$$\boldsymbol{\phi}_c = \sum_{i \in c} \frac{p(c)_i}{N} \ , \qquad \boldsymbol{\mu}_c = \frac{\sum_{i \in c} p(c)_i \mathbf{z}_i}{\sum_{i \in c} p(c)_i} \ , \tag{6.4}$$

$$\boldsymbol{\Sigma}_c = \frac{\sum_{i \in c} p(c)_i (\mathbf{z}_i - \boldsymbol{\mu}_c)(\mathbf{z}_i - \boldsymbol{\mu}_c)^T}{\sum_{i \in c} p(c)_i} \ . \tag{6.5}$$

The energy function of each class is inferred by the likelihood with their estimated parameters

$$E(\mathbf{z}_i) = \boldsymbol{\phi}_c \mathcal{N}(\mathbf{z}_i; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \ . \tag{6.6}$$

Consequently, the maximum likelihood estimation of the parameters can be reduced to the maximum point of $E(\mathbf{z})$. Since the logarithmic function is a monotonically increasing function, the log-likelihood $\log E(\mathbf{z})$ has the same maximum value as $E(\mathbf{z})$. Moreover, in many cases, it is relatively simple to find the maximum value of the logarithmic function, so the maximum point of the likelihood is usually changed to the maximum point of the log-likelihood.

For measuring the similarity between the energy and truth class label y, it is hoped that the likelihood function of $\mathbf{z}$ reaches a maximum or the negative log-likelihood $-\log E(\mathbf{z})$ reaches a minimum, when $\mathbf{z}$ belongs to the y class. However, when $\mathbf{z}$ does not belong to class y, we should minimise $\log E(\mathbf{z})$. Since $\log E(\mathbf{z})$ does not have a minimum, the natural loss function (cross entropy) is not applicable. Here we introduce $p(c)$ whose value is within $[0, 1]$, and the value of $\log E(\mathbf{z})$ is proportional to $p(c)$. We then minimise the loss function by leveraging the minimum of these two terms. Then the loss function of the classification network (Categorical losses) is inferred as

$$L_c(E(\mathbf{z}), y) = -\sum_{c=1}^{C} y \log E(\mathbf{z}) + \sum_{c=1}^{C} (1-y) \log \big(1 - p(c)\big) \log E(\mathbf{z}) \ . \tag{6.7}$$

Besides, the regularisation $P(\boldsymbol{\Sigma}_c) = \sum_{c=1}^{C} \sum_{j=1}^{d} (\hat{\boldsymbol{\Sigma}}_{cjj})^{-1}$ solves the singularity problem by penalising small values on the diagonal entries, where d is the low-dimensional representations' dimensions. Given the above, this objective function can be constructed as follows

$$J(\boldsymbol{\theta}_c, \boldsymbol{\theta}_d, \boldsymbol{\theta}_p) = \frac{1}{N} \sum_{i=1}^{N} L_{AE}(\mathbf{x}_i, \mathbf{x}_i') + \frac{\lambda_1}{N} \sum_{i=1}^{N} L_c\big(E(\mathbf{z}_i), y_i\big) + \frac{\lambda_2}{N} P(\hat{\boldsymbol{\Sigma}}_c), \tag{6.8}$$

where λ_1 and λ_2 are the meta parameters. Furthermore, the class label can be predicted with the highest energy $E(\mathbf{z})$ of the test image among all classes.

6.1.2 Optimisation Strategy

From Fig. 6.2, the proposed network consists of two sub-networks, including deep AE-based generation network and the classification network. Through experiments, we found that the gradient direction is difficult to obtain during the initial training, which makes the network converge slowly. One reason is that the generation network and the classification network may have conflicted objectives, i.e., the optimisation objective of the deep AE is to minimise the difference between the generated and the original images, while the optimisation objective of the classification network is to estimate the distribution of each category which could push further the different classes. Moreover, the parameters of the likelihood function are usually sensitive to initialisation. However, the parameters of the density estimation are randomly initialised by the encoder network in the proposed network. In order to solve the problem, we adopt an optimisation strategy as shown in Algorithm 5. The algorithm intends to optimise the classification network M epochs first, and then the generator is involved for optimisation. Here, M is set to a number that is not big enough for the classifier to converge (with the validation accuracy around 60%). It is worth mentioning that this optimisation strategy is only used to enable the network to find a good optimisation direction quickly. In terms of results, there is an only minor improvement by using this strategy.

Algorithm 5: Optimisation procedure for AE-based joint learning. M is the number of epochs to update the parameter of the density network. η denotes the learning rate.

```
Input        : A training set $[\mathbf{x}_1, \dots, \mathbf{x}_D]$.
Density Model: $L_c(E(\mathbf{z}_i), y; \boldsymbol{\theta}_d)$.
Whole Model  : $J(\boldsymbol{\Phi})$, $\boldsymbol{\Phi} = \{\boldsymbol{\theta}_e, \boldsymbol{\theta}_c, \boldsymbol{\theta}_d, \boldsymbol{\theta}_p\}$.
for $k = 1$ to $M - 1$ do
    Update $\boldsymbol{\theta}_d$,
        $\boldsymbol{\theta}_d^{k+1} \leftarrow \boldsymbol{\theta}_d^k + \eta \nabla_{\boldsymbol{\theta}_c^k} L_c(E(\mathbf{z}_i), y; \boldsymbol{\theta}_d^k)$.
    Update $\boldsymbol{\theta}_p$,
        $\boldsymbol{\theta}_p^{k+1} \leftarrow \boldsymbol{\theta}_p^k + \eta \nabla_{\boldsymbol{\theta}_p^k} P(\hat{\boldsymbol{\Sigma}}_c, \boldsymbol{\theta}_p^k)$.
for $k = M$ to $K$ do
    Update all parameters,
        $\boldsymbol{\Phi}^{k+1} \leftarrow \Phi^k + \eta \nabla_{\boldsymbol{\Phi}^k} J(\boldsymbol{\Phi}^k)$
```

6.2 Preliminary Experimental Results

We mainly design two parts of experiments to validate the proposed model in both a qualitative and quantitative way. First, we will give the qualitative results on the handwritten digits dataset MNIST, including the reconstruction results compared with several deep neural networks, and the generated result compared with the deep density model shown in Chapter 5. Then, we conduct quantitative experiments on a variety of datasets to evaluate the classification error rate, including three handwritten datasets and one $S3$ dataset. The experimental results will be compared with 2L-MJFA which is shown in Chapter 3.3. Also, the rejection results on handwritten characters are reported and compared with the convolutional neural network (CNN).

6.2.1 Datasets

The following datasets are utilised in our quantitative experiments.

- ULC: The urban land cover (ULC) data are used to classify high-resolution aerial images which consists of 9 types with 273 training samples, 77 test samples and 147 attributes [106][107].

- MNIST: This handwriting digit data contain 0 to 9 digits images of size 28 by 28 pixels [169]. Each image is reshaped to a 784-dimensional vector for two provirus work. The training set includes $1,000$ samples of each digit, and the test set also consists of $1,000$ of each digit.

- CASIA-HWDB: The off-line Chinese handwriting character dataset contains $3,755$ classes of size 64×64 images. The training set includes around 236 samples of each Chinese character and the test set also consists of around 60 of each Chinese character [178]. To facilitate the 2L-MJFA model, we used a benchmark feature extraction method 8-direction histogram feature extraction combined with pseudo-$2D$ bi-moment normalisation for representing a character sample. The feature dimensionality is further reduced to 160 by FDA [179].
  - HZ-20: The extracted data of the first 20 classes.
  - HZ-297: The extracted data of the first 297 classes.

6.2.2 Network Configuration

The network structures of the proposed deep autoencoder-based joint learning model (AE-based JL) are summarised for the individual datasets as follows.

- The network structure employed on ULC dataset is presented as follows.

  The deep AE network runs with

  $FC(147, 64, tanh) - FC(32, 4, tanh) - FC(64, 12, tanh) - FC(12, 10, none) -$
  $FC(10, 12, tanh) - FC(12, 64, tanh) - FC(64, 147, none)$,

  and the GMM network performs with

  $FC(10, 10, tanh) - Drop(0.5) - FC(10, 20, tanh)$

  $-Drop(0.5) - FC(20, 9, softmax)$.

- The network structure employed on MNIST dataset is presented as follows.

  The deep AE network runs with

  $CONV(1, 28, c(5, 2, 1), BatchNorm2d(28), LeakyReLU(0.2))-$

  $CONV(28, 64, c(3, 2, 1), BatchNorm2d(64), LeakyReLU(0.2))-$

  $CONV(64, 128, c(3, 2, 1), BatchNorm2d(128), LeakyReLU(0.2))-$

  $FC(128 * 16, 32, tanh) - FC(32, 4, none)-$

  $FC(4, 32, tanh) - FC(32, 128 * 16, BatchNorm2d(128))-$

  $DCONV(128, 64, c(3, 2, 1), BatchNorm2d(64), ReLU(0.2))-$

  $DCONV(64, 28, c(3, 2, 1), BatchNorm2d(28), ReLU(0.2))-$

  $DCONV(28, 1, c(5, 2, 1), Sigmoid)$,

  and the GMM network performs with

  $FC(4, 10, tanh) - Drop(0.5) - FC(10, 20, tanh) - Drop(0.5) - FC(20, 10, softmax)$.

- The network structures employed on HZ-20 and HZ-297 dataset are presented as follows.

  The deep AE network runs with

  $CONV(1, 64, c(3, 2, 1), BatchNorm2d(64), LeakyReLU(0.2))-$

  $CONV(64, 128, c(3, 2, 1), BatchNorm2d(128), LeakyReLU(0.2))-$

  $CONV(128, 256, c(3, 2, 1), BatchNorm2d(256), LeakyReLU(0.2))-$

  $CONV(256, 256, c(3, 2, 1), BatchNorm2d(256), LeakyReLU(0.2))-$

  $FC(256 * 16, 32, tanh) - FC(32, 16, none)-$

  $FC(16, 32, tanh) - FC(32, 256 * 16, BatchNorm2d(256))-$

  $DCONV(256, 256, c(3, 2, 1), BatchNorm2d(256), ReLU(0.2))-$

$DCONV(256, 128, c(3, 2, 1), BatchNorm2d(128), ReLU(0.2))-$

$DCONV(128, 64, c(3, 2, 1), BatchNorm2d(64), ReLU(0.2))-$

$DCONV(64, 28, c(3, 2, 1), BatchNorm2d(28), ReLU(0.2))-$

$DCONV(28, 1, c(3, 2, 1), Sigmoid),$

and the GMM network performs with

$FC(16, 64, tanh) - Drop(0.5) - FC(64, 128, tanh) - Drop(0.5)-$

$FC(128, 128(256), tanh) - Drop(0.5) - FC(128(256), 20(297), softmax).$

Here $CONV/DECONV(a, b, c(kernel_size, stride, padding), d, f)$ means a convolutional/ deconvolutional layer with a input neurons and b output neurons activated by function f, $FC(a, b, f)$ means a fully-connected layer with a input neurons and b output neurons activated by function f (none means no activation function is exploited), and $Drop(p)$ denotes a dropout layer with keep probability p during training. In practice, the settings of the meta parameters are $\lambda_1 = 0.1$, $\lambda_2 = 0.005$ for generation and $\lambda_1 = 0.5$, $\lambda_2 = 0.005$ for classification.

It is worth noting that we only selected standard autoencoder and convolution networks as the compression network in the preliminary experimental. Other better performing networks, such as DenseNet or ResNet, can be used as alternative compression networks.

6.2.3 Benchmarking Approaches

The network structures of the competitive methods are listed in following. The competitive methods on reconstruction and rejection experiments.

- Deep autoencoder - Adopting the same deep AE network of the AE-based JL model employed on MNIST dataset.

- Convolutional neural network (reconstruction):

  $CONV(1, 28, c(5, 2, 1), BatchNorm2d(28), LeakyReLU(0.2))-$

  $CONV(28, 64, c(3, 2, 1), BatchNorm2d(64), LeakyReLU(0.2))-$

  $CONV(64, 128, c(3, 2, 1), BatchNorm2d(128), LeakyReLU(0.2))-$

  $FC(128 * 16, 32, tanh) - FC(32, 20, tanh) - Drop(0.5) - FC(20, 20, tanh)-$

  $Drop(0.5) - FC(20, 10, softmax).$

- Joint learning the deep autoencoder and convolutional neural network - The same network structures of AE-based JL model on the MNIST dataset is adopted, but there is no process of density estimation after it.

- Convolutional neural network (rejection):

  $CONV(1, 64, c(3, 2, 1), BatchNorm2d(64), LeakyReLU(0.2))-$

  $CONV(64, 128, c(3, 2, 1), BatchNorm2d(128), LeakyReLU(0.2))-$

  $CONV(128, 256, c(3, 2, 1), BatchNorm2d(256), LeakyReLU(0.2))-$

  $CONV(256, 256, c(3, 2, 1), BatchNorm2d(256), LeakyReLU(0.2))-$

  $FC(256 * 16, 64, tanh) - Drop(0.5) - FC(64, 128, tanh) - Drop(0.5)-$

  $FC(128, 128, tanh) - Drop(0.5) - FC(128, 20, softmax).$

The competitive methods on generation and classification experiments have been described in detail in the previous sections: 2L-MJFA have been shown in Chapter 3.3.3,DM-FA have been shown in Chapter 2.4.2, and DMCFA have been shown in Chapter 5.1.

6.2.4 Qualitative Results

The experiments presented in this section only demonstrate qualitative results. First, we engage the public benchmark data MNIST to demonstrate the performance of the proposed model in feature extraction. Then, we measure the ability of the AE-based joint learning model as a generative model to capture the data distribution by comparing the generated images on the MNIST.

Empirical Results

Figure 6.3 demonstrates the 2-D hidden layers (latent features) yielded by the proposed AE-based JL network, the deep autoencoder (AE) network, the convolutional neural network (CNN), joint learning the deep autoencoder and convolutional neural network (AE-CNN), separately. In these qualitative experiments, the classification accuracies of CNN, AE-CNN, and AE-based joint learning are 96%, 97%, and 97%, respectively. Note that we reduced the latent feature space to 2-dimension for better illustration on AE-based JL, AE and AE-CNN, and set two neurons in the penultimate layer of the CNN, leading their performance not as high as the state-of-the-art.

The visualised points of AE and our proposed network are from the output layers of the encoder, and the visualised points for CNN are from the previous layer of the class output layer. The results of the AE show that the digits do not exhibit significant boundaries in the low-dimensional feature space without the guidance of the category information. As shown in Fig. 6.3(a), the feature points of 9, 4, and 7 are mixed together, leading that 9 is restored as 4 or 7 in reconstruction (see the results of AE in 8th and 10th columns of Fig. 6.4).

Fig. 6.3(b) shows the low-dimensional feature space of CNN. The results in two-dimensional space show that, though the classification boundary is well obtained, the distance between classes is not pushed away in this feature space. However, by learning AE and CNN jointly, the distance of each category is pushed further in the feature space, as seen in Fig. 6.3(c) .

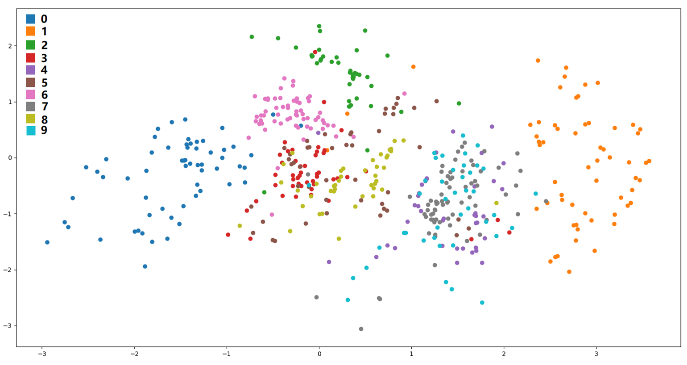
For AE-based JL, the hidden layer is encoded into a 2-D Gaussian distribution. The two-dimensional representations are shown in Fig. 6.3(d). In this low-dimensional feature space, 3, 5, and 8 are relatively close; in fact, they are similar in writing. After the category information is used, 7 and 9 are completely separated, and the results of AE-CNN are the same. From the shape of each type of cluster, they all conform to the shape of a two-dimensional Gaussian distribution. It is clear to see that the latent features of each class can be aggregated as a Gaussian distribution and the label information can better regularise the hidden code.

Fig. 6.4 demonstrates the reconstructed hand-writing digits of the AE-based JL, AE, and AE-CNN. The first row shows ground-truth images, and each of the remaining rows shows the reconstructed images give by each different method. The hidden code z is fixedly reduced to 2-dimension. Also, the category information is utilised to guide the encoder except for the AE systematically. The results of AE imply that it can improve the accuracy of the reconstruction by using the classification information.

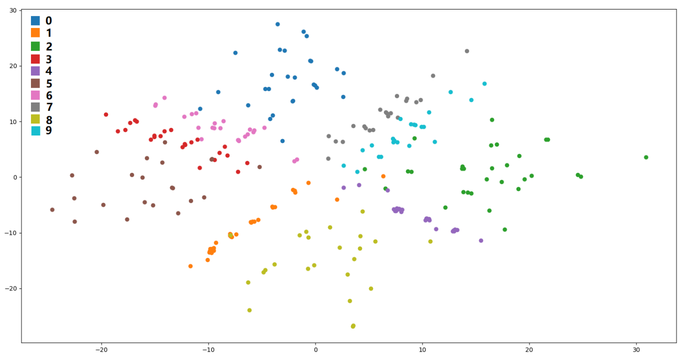
The difference between AE-based JL and AE-CNN is that the former method expects each class to conform to a Gaussian distribution. The reconstructed images by AE-CNN are indistinct and dedicated to restoring each type of number to the same shape. However, AE-based JL is able to generate high-quality digits which are closer to the ground-truth.

Generation Results

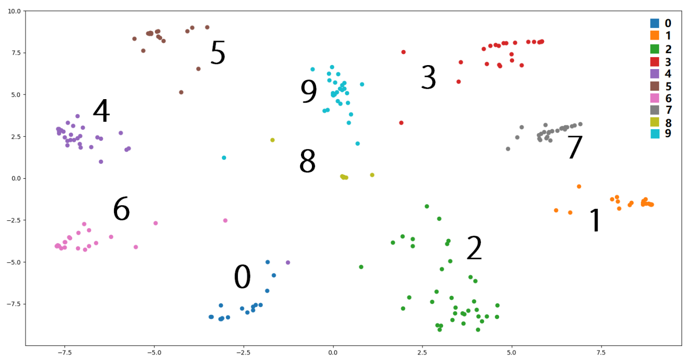
In the second set of this experiment, we engage one example to demonstrate the generation result of the deep autoencoder-based joint learning network learned by the end-to-end training, compared with two deep density models that rely on the Deep Belief Network. All the three methods can generate new samples with low-dimensional features which are sampled with the mean and variance of each class. The results of the generation are shown in Fig. 6.5. In the figure, each block presents a digit, including $0 - 9$. In each block, the first two rows present the results of Deep MFA and Deep MCFA (details are shown in Chapter 5), which are models based on the DBN framework with the layer-wise algorithm. The last two rows show the results of AE-based JL using different meta parameter, $\lambda_1 = 0.1$ or 0.5. When the meta parameter λ_1 is increased, the impact of the category information is enlarged. As expected, the last two rows' results show that, when the classification sub-network is enhanced, the clarity of the generated samples will



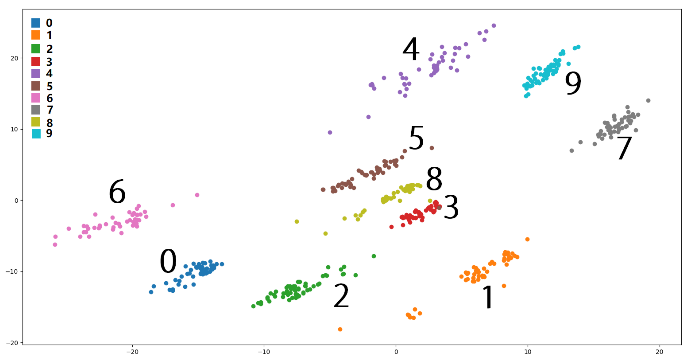
(a)



(b)

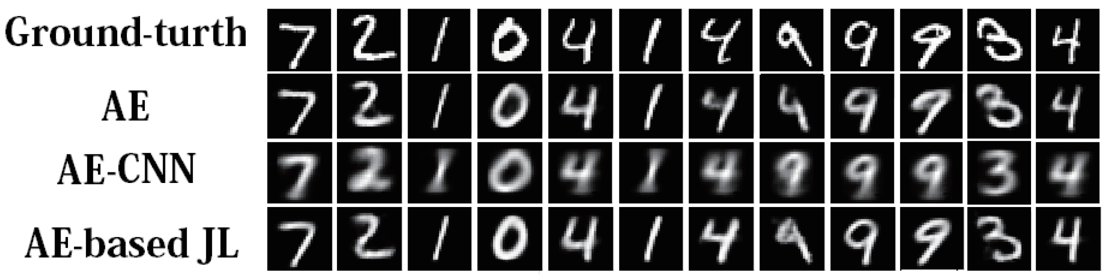


(c)



(d)

Fig. 6.3: The sketch of classification and dimensionality reduction. From top to bottom: the 2-dimensional representation of the features by using (a) the deep autoencoder model, (b) the convolutional neural network, (c) the deep autoencoding convolutional neural network (AE-CNN) and (d) the AE-based joint learning model, where each colour denotes a class.



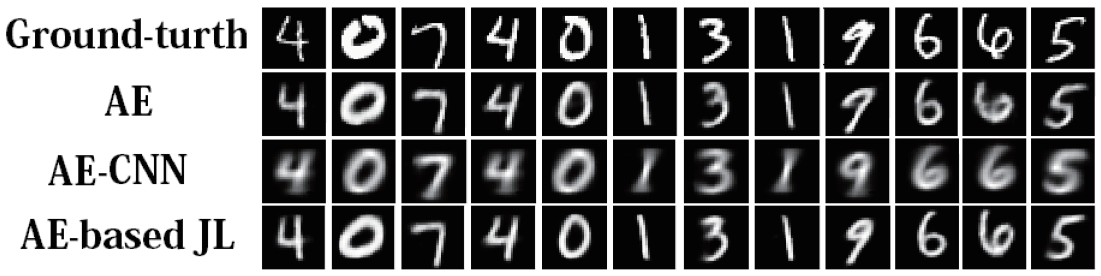


Fig. 6.4: Reconstruction results on the testing set of MNIST. In each block, the top row shows the ground-truth images; the second row shows the reconstructed images by deep autoencoder which has the same structure as the AE part of the AE-based joint learning; the third row shows the reconstructed images by learning the deep autoencoder and CNN jointly, where the AE structure is also the same as that of the AE-based joint learning; the bottom row demonstrates the reconstruction images by the AE-based joint learning.

be reduced, but the impact on diversity is not obvious. In Chapter 5.3.3, the generation results of the deep density models are clearer than their shallow forms. In this work, from the comparison, AE-based JL demonstrates superior performance over the other two deep density models in both the diversity and quality.

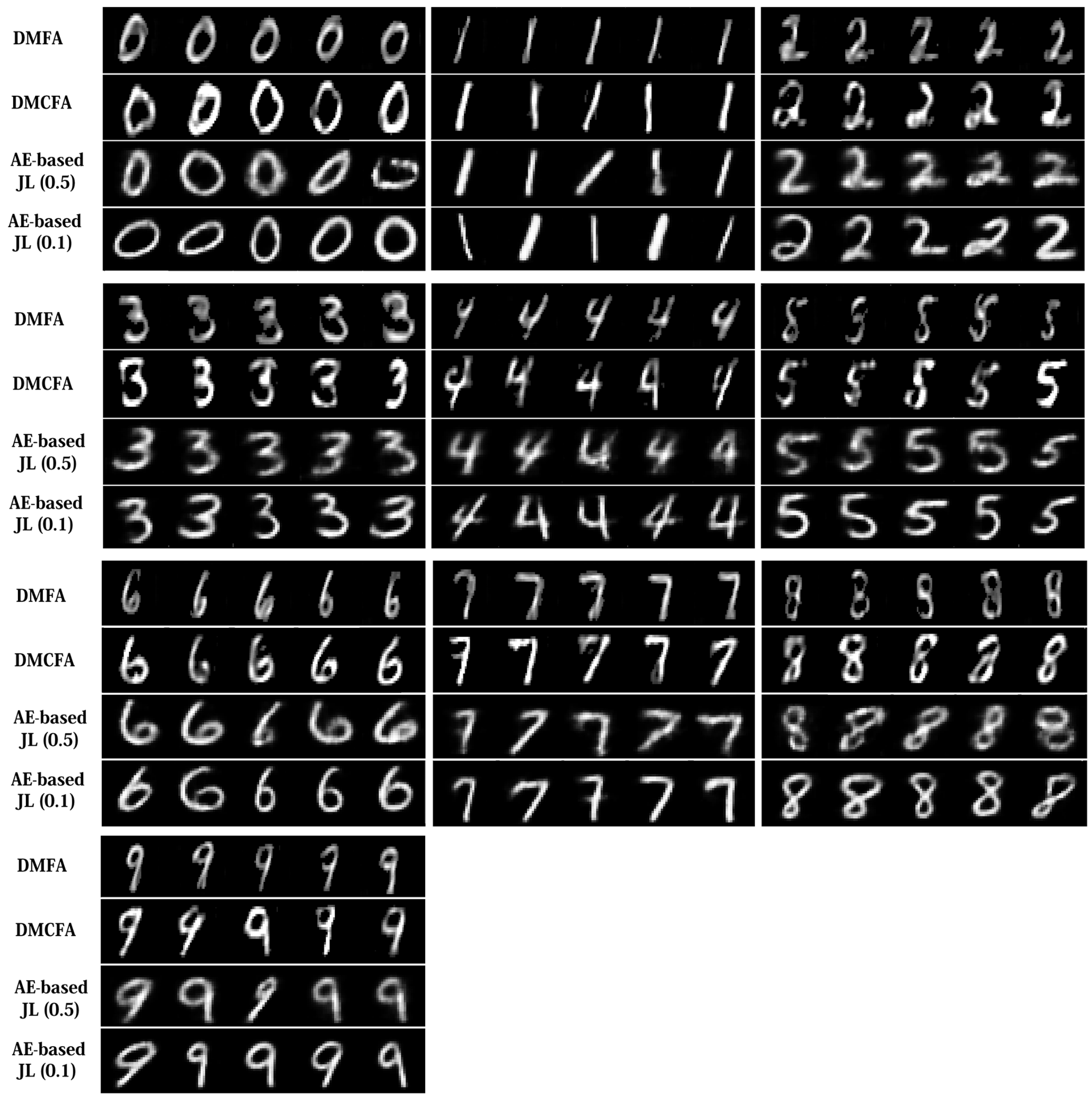


Fig. 6.5: Generation results of the handwriting digits. In each block, the top to bottom rows show the generative digits of 2-layer DMCFA, 2-layer DMFA, AE-based joint learning model with $\lambda_1 = 0.5$, and $\lambda_1 = 0.1$, separately.

6.2.5 Quantitative Results

For the quantitative result, we design two experiments: the comparison with the 2L-MJFA (shown in Chapter 3.3) on classification, and the comparison with CNN on rejection.

Classification

We report the classification accuracy (error rate as the metric) by comparing our proposed work, AE-based joint learning (AE-JL) and our previous work 2L-MJFA. The results on 4 real-world datasets are shown in Table 6.1. The result shows that the mixture joint learning methods 2L-MJFA provides lower error rates on the S3 dataset (ULC dataset). The proposed network gets 99.4% accuracy on the training set of the S3 dataset. But it just gets 55.63% accuracy on the test set, which is mainly caused by over-fitting due to the insufficient number of observed samples. Therefore, 2L-MJFA is a better choice when the samples are insufficient.

On the MNIST data, the AE-based JL outperforms the 2L-MJFA. Note that, there are $10,000$ samples for training and $10,000$ samples for testing, where each digit has $1,000$ training samples. From the results, it is unwise to search for a low-dimensional representation space by using just one multi-linear projection when working on the image data. In addition, when there are many categories, it is difficult to find the optimal low-dimensional feature space only by using multi-linear mapping. To better illustrate the performance, we also give the results of 2L-MJFA classification in a small number of categories. When the number of categories is randomly taken at 2, 4, 6 and 8, the results are 0.0013, 0.0799, 0.1035 and 0.1055, respectively.

In the latter two sets of Chinese character handwritten recognition experiments, the features used by 2L-MJFA are the 8-direction histogram features extracted from the character images, while the AE-based JL is directly trained on raw data features. In particular, when the dimensionality is reduced to 140 (the actual dimension), 2L-MJFA yields the best performance with the error rates being 0.04598 and 0.1012. During the experiment, the network structure and meta parameters setting on HZ-20 and HZ-297 are the same. When the hidden code z is fixedly reduced to 16-dimension, the error rates is 0.04682 and 0.1978. The insufficient samples should be the main reason that limits its performance of the deep network.

Rejection

The results of the rejection shown in this experiment are based on a multi-classifier for handwritten character recognition. The general definition of rejection is to determine the confidence of the recognition results according to a certain strategy when the classifier

Table 6.1: Performance on 4 real-world datasets in terms of the error rate (the smaller, the better).

—Error Rate—				
Dataset	ULC	MNIST	HZ-20	HZ-297
2L-MJFA	**0.1099**	0.2100	**0.04598**	**0.1012**
AE-based JL	0.4437	**0.0324**	0.04682	0.1978

recognises the unknown sample. This strategy is called a rejection mechanism. If the rejection mechanism considers the recognition results to be untrustworthy, the learning system rejects to recognise the samples and indicates that the recognition fails; otherwise, the recognition results are employed. If the learning system does not adopt the rejection mechanism, it degenerates into a standard classifier and directly returns the recognition results.

The first step in rejection is to train the classifier. In this experiment, two classifiers, the AE-based JL, and CNN, are trained on the handwriting digits datasets (the training set of MNIST), and both of them achieved 99% accuracy in the test dataset. Then, we created two datasets for the rejection task. The first dataset is to add 600 handwritten letter samples (from the MNIST test set) in $1,000$ handwritten digit samples (from MNIST test set). Another is to add $1,128$ Chinese handwritten character samples (the test set of HZ-20) in $1,000$ MNIST test samples. Here, we set the test set of MNIST as normal-samples, and the other test set is used for rejection is rejection-samples. Finally, the rejection mechanism of this experiment is directly set by a threshold of the return result. The learning system returns a rejection when the result is greater than the threshold. The threshold of the AE-based JL is set on the value of energy function ($E(\mathbf{z})$), and CNN's is set on the output results after softmax. CNN returns a rejection when the result is less than the threshold.

The criteria used to measure the rejection results are established as follows.

- Ground-truth Positive (GTP): The ground-truth of the normal-samples.
- Ground-truth Negative (GTN): The ground-truth of the rejection-samples.
- True Positive Rate (TPR): True positive (TP) means that the test sample is correctly classified as the normal-sample. The true positive rate is calculated as $TPR = TP/GTP$.
- True Negative Rate (TNR): True negative (FN) means that the test sample is classi-

Table 6.2: Rejection results regarding three metrics. For TPR and TNR, the bigger, the better. For TER, the smaller, the better. There are 600 rejection-samples with letters, $1,128$ rejection-samples with Chinese characters, and $1,000$ normal-samples.

—Rejection Results—					
Dataset	MNIST	EMNIST-Letter		Chinese Characters	
Criteria	TPR	TNR	TER	TNR	TER
AE-based JL	0.9700	**0.6217**	**0.1600**	**0.8528**	**0.0921**
CNN	0.9600	0.5017	0.2119	0.6773	0.1898

fied as the rejection-sample, and their ground-truths are the rejection-samples. The true negative rate is calculated as $TNR = TN/GTN$.

- Total Error Rate (TER): Error results on all test dataset, including the misclassified normal-samples (FP) and the rejection-samples that are detected as normal-samples (FN). The total error rate is calculated as $TER = (FP + FN)/(GFP + GTN)$.

The results of rejection are shown in Table 6.2. For the AE-based JL, the threshold is set to 9.999×10^8. When $E(\mathbf{z}) > 9.999 \times 10^8$, the sample will be rejected. For the CNN, the sample will be rejected when the output result is less than 99%, The first column in the table presents the true positive rate. This rate actually equals the accuracy after considering the threshold. Compared with the test accuracy, the recognition values of both the methods are reduced to 97% and 96%, separately. This suggests that the confidences of some correct classification results are not high enough. From the EMNIST-Letter results, although our proposed method is superior to CNN in both metrics, the results are still not very good.

To better illustrate the performance, we also plot the recognition results and the raw images of the first 120 handwritten letters in Fig. 6.6, where it can be observed that many letters are similar to the handwriting of digits, such as Z and 2, O and 0, S and 5. In this figure, the red labels R denote the rejections, and the black labels show the recognised results. Obviously, the true negative rate is much higher on Chinese characters in our proposed method, which also outperforms CNN. Note that this experiment is still quite preliminary. The performance can be further improved if a threshold is set for each category.

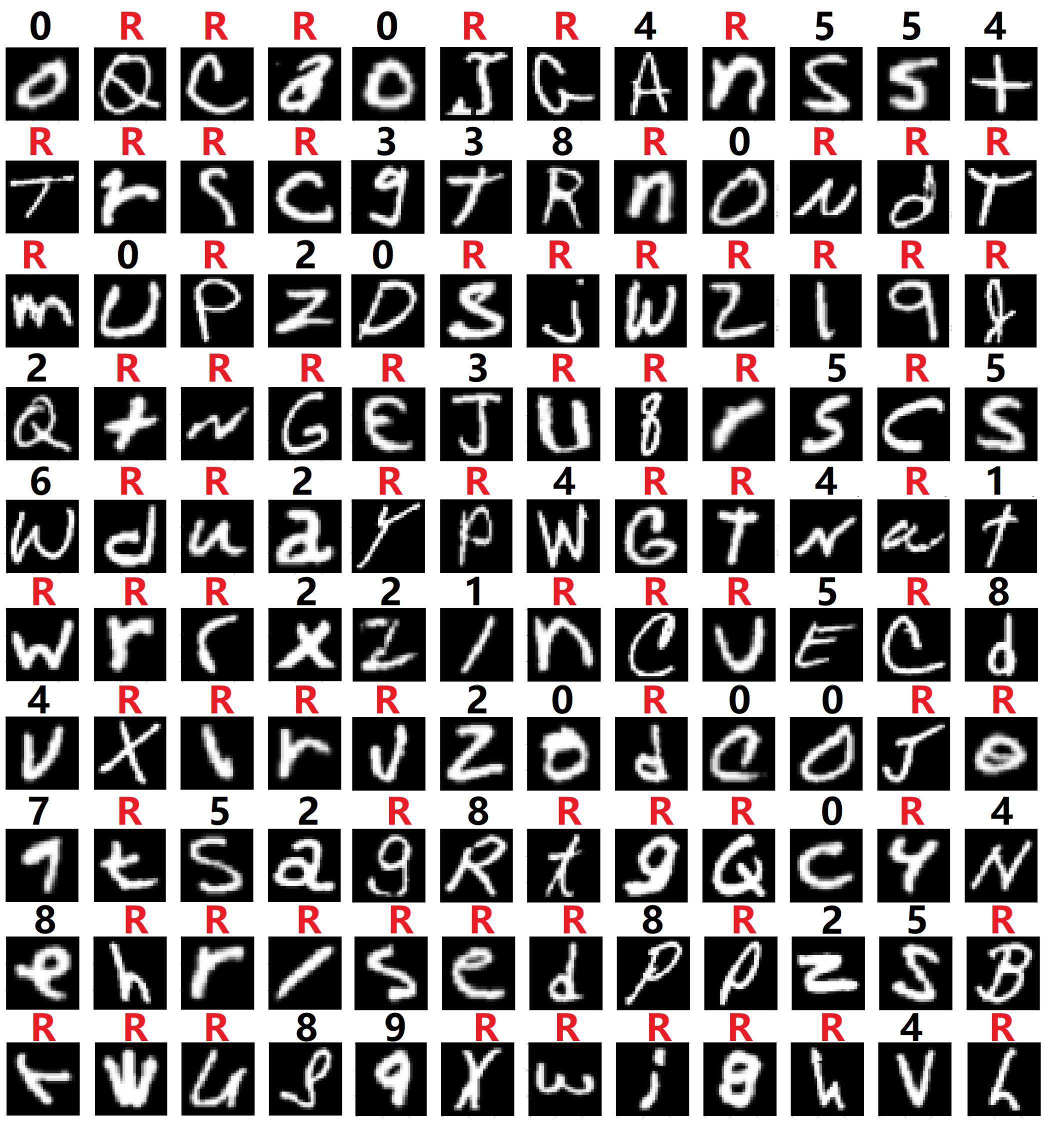


Fig. 6.6: Rejection results of 120 handwriting letters. The red R denotes the rejected results.

6.3 Discussion and Future Work

In this chapter, we proposed a novel Deep Neural Network-Based Joint Learning Model, which well combines Deep Autoencoder and Density Model.

The contributions of this method can be summarised as follows. (1) The generation and the classification networks are learned simultaneously; (2) it is trained in an end-to-end fashion that does not require pre-training of the autoencoder network, and (3) the parameter optimisation is more straightforward than the previous methods. However, the singular matrix problem caused by the density estimation still needs to be solved. To this end, how to embed the density estimation into deep neural networks through the representation of some hidden layers will be the focus of our future work. Moreover, further applications can be extended to, such as anomaly detection and out-of-distribution detection. Finally, how to embed the infinite mixture model into a deep neural network is also one of our concerned topics.

Chapter 7

Conclusion and Future Work

In this chapter, the conclusion of this thesis is provided. First, the whole journey of this dissertation will be briefly reviewed, which starts from the challenges of density learning in the literature. Motivated from these challenges, we study three types of structured latent variables including the finite latent variables, the infinite latent features, and deep-structured latent variables. These study eventually leads to different models as discussed in this thesis. Following that, we then present future perspectives within the intended framework.

7.1 Review of the Journey

This dissertation is focused on learning density models with structured latent variables typically from high-dimensional data. In the literature, five major how-to concerns are outlined: how to capture the crucial attributes in lower-dimensional space, how to reduce the free parameters, how to estimate distributions to fit the complicated data manifold, how to increase model flexibility, and how to reduce the learning difficulties. In this dissertation, these challenges are managed by constructing the different latent variable structures of density models. The structured latent variables can correspond to different concepts by assuming the different density distributions on latent variables. We have reviewed the state-of-the-art topics in the areas of finite mixture models, infinite latent features models, and deep models. More importantly, we have made our contributions in each of these topics.

Based on the finite latent variable structures, we have introduced several joint learning models which are further applied in both clustering and classification. First, by designing a common loading matrix, a finite mixture model is proposed managing to perform learning with dimensionality reduction simultaneously. It is noted that the common loading matrix can be regarded as a global dimensionality reduction matrix, making the effective low-dimensionality representations can be captured and calibrated for the subsequent learning tasks. One additional advantage is that the proposed model can reduce significantly the free parameters as used in the traditional finite mixture model. Then, we propose a two-layer mixture model with a global loading matrix for discriminant analysis. This is a mixture of mixtures structure which is used to capture the complex properties of each class better. The approach has been validated on both synthetic datasets and real-world datasets including the benchmark clustering datasets and the small sample size datasets. The performance of the proposed joint learning models is demonstrated to significantly outperform the separated learning models in clustering. Also, this two-layer mixture model with a global loading matrix leads to the best results compared with other mixture component models in classification, when the sample numbers of each class are limited.

In order to address the limitation that certain parameters need to be pre-specified in many finite mixture models, an infinite latent feature model is proposed. The non-parametric prior (IBP prior) is involved for improving the flexibility of the infinite model, which can automatically determine an optimal number of features. Meanwhile, we have contributed a tri-factorisation framework to reveal the latent structures among items (samples) and attributes (features). This model also delivers latent binary features needing no extra constraints. An efficient optimisation algorithm is also developed accordingly. In the experiments, the proposed infinite latent feature model significantly outperforms the other four competitive algorithms on various tasks including reconstruction, pre-image restoration, and clustering. Moreover, a series of experiments on feature extraction have demonstrated that the proposed tri-factorisation model has superior ability to extract both the latent structures and the features particularly from the data with complex structures.

Finally, we have introduced two models via deep-structured latent variables: a layer-wise-based model, and a deep autoencoder-based model. Both the deep models are proposed with the purpose of fitting the complicated data manifold as well as alleviating the learning difficulty. The first deep density model adopts a greedy layer-wise learning approach exploiting the same scheme to train each hidden layer. Its inference and parameter computation procedure are more straightforward than previous methods. This model is evaluated on empirical tasks including clustering and generation on real-world datasets. The results show that the proposed model achieves better performance compared with those standard and state-of-the-art layer-wise-based methods. The other deep density model employs a deep autoencoder as a dimensionality reduction procedure and tries to optimise the parameters jointly. The deep autoencoder is more powerful than a simple multi-linear projection in finding low-dimensional representations. Importantly, this is an end-to-end model that does not require the pre-training of the autoencoder network. It can also be straightforwardly optimised with the standard backpropagation. In the experiment, this model was evaluated on reconstruction, generation, classification, and rejection tasks. The results show that, in the case of insufficient data, our previous works are more applicable. In addition, the proposed model has demonstrated outstanding performance compared with the general deep autoencoder and convolutional neural network.

7.2 Future Work

Although several contributions have been made in solving the above problems, there is still much room for further improvement.

First, despite their excellent expressive ability, deep networks usually need to specify many parameters beforehand. To determine these parameters automatically, infinite deep density model involving non-parametric prior might be a possible option.

Second, the layer-wise-based model is closely related to distribution estimation of the convolutional kernels, which is called deep dictionary learning. Our next future work is to perform deep dictionary learning and classifier design jointly.

Third, to further extend the deep autoencoder-based model, we intend to study how the parameters of Gaussian distribution can be represented by hidden layers rather than a sub-network. This would avoid certain computational problems and can lead to more stable and simple network structures.

Finally, how to build up adaptive neural networks is also an interesting problem. When new categories of data are fed, a good learning system should be able to first detect these abnormalities and then adapt its network rather than having to retrain the network. We will also look into this issue in the future.

Publication List

Here is the publication list during my Ph.D. study:

1. Xi Yang, Kaizhu Huang, Rui Zhang, John Y. Goulermas, "A Novel Deep Density Model for Unsupervised Learning," Cognitive Computation, pp. $1-11$, 2018.

2. Xi Yang, Kaizhu Huang, Rui Zhang, John Y. Goulermas, Amir Hussain, "A New Two-layer Mixture of Factor Analyzers with Joint Factor Loading Model for the Classification of Small Dataset Problems," Neurocomputing, vol. 312, pp. $352-363$, 2018.

3. Xi Yang, Kaizhu Huang, Rui Zhang, Amir Hussain, "Learning Latent Features with Infinite Non-negative Binary Matrix Tri-factorisation," IEEE Transactions on E-merging Topics in Computational Intelligence, vol. 2, no. 3, 2018.

4. Xi Yang, Kaizhu Huang, John Y. Goulermas, Rui Zhang, "Joint Learning of Unsupervised Dimensionality Reduction and Gaussian Mixture Model," Neural Processing Letters, vol. 45, no. 3, pp. $791-806$, 2017.

5. Xi Yang, Kaizhu Huang, Rui Zhang, "Deep Mixtures of Factor Analyzers with Common Loadings: A Novel Deep Generative Approach to Clustering," in Proceedings of International Conference on Neural Information Processing, Springer, 2017, pp. $709-719$.

6. Xi Yang, Kaizhu Huang, Rui Zhang, Amir Hussain, "Learning Latent Features with Infinite Non-negative Binary Matrix Tri-factorisation," in Proceedings of International Conference on Neural Information Processing, Springer, 2016, pp. $587-596$.

7. Xi Yang, Kaizhu Huang, Rui Zhang, John Y. Goulermas, "Two-layer Mixture of Factor Analysers with Joint Factor Loading," in Proceedings of International Joint Conference on Neural Networks, IEEE, 2015, pp. $1-8$

8. Xi Yang, Kaizhu Huang, Rui Zhang, "Unsupervised Dimensionality Reduction for Gaussian Mixture Model," in Proceedings of International Conference on Neural Information Processing, Springer, 2014, pp. $84-92$.

Appendix

Here is the real-world dataset list adopted in this dissertation:

1. User Knowledge Modeling Dataset [119]: It is the real dataset about the students' knowledge status about the subject of Electrical DC Machines. This dataset consists of 4 knowledge levels of the students with 403 training samples, 206 test samples. and 5 attributes.

2. Physical Dataset [103]: These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the 3 types of wines. This dataset consists of 178 training samples and 13 attributes.

3. Iris Dataset [180]: It is the real dataset about the iris plant. The dataset contains 3 classes of 50 instances each and 4 attributes, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are not linearly separable from each other.

4. Seeds Dataset [181]: The examined group comprised kernels belonging to 3 different varieties of wheat: Kama, Rosa and Canadian, 70 elements with 7 attributes each, randomly selected for the experiment. High quality visualization of the internal kernel structure was detected using a soft X-ray technique. It is non-destructive and considerably cheaper than other more sophisticated imaging techniques like s-canning microscopy or laser technology. The images were recorded on 13×18 cm X-ray KODAK plates. Studies were conducted using combine harvested wheat grain originating from experimental fields, explored at the Institute of Agrophysics of the Polish Academy of Sciences in Lublin.

5. Breast Cancer Wisconsin Dataset [120, 121]: This dataset contains two subsets, the Wisconsin diagnostic breast cancer (WDBC) and the Wisconsin prognostic breast cancer (WPBC). WDBC contains 569 instances which are divided into the two diagnostic predictions of benign and malignant. The 60 attributes consist of 30

real-valued input features. WPBC contains 194 instances, which record two classes of patients, that is being recurrent or not post-surgical.

6. LSVT Dataset [122]: The LSVT Voice Rehabilitation Data Set (LSVT) contains 98 instances with 309 attributes and is used for evaluating whether a phonation considered acceptable or not after voice rehabilitation (2 classes classification problem) [122].

7. Breast Tissue Dataset [123]: This dataset contains 106 objects described by 9 features. For each object, a group of features are selected from excised breast tissue samples using electrical impedance measurement. 6 major diagnostic classes are involved that consist of 4 normal breast tissues: connective, glandular, Fibroadenoma and adipose tissue, as well as 2 pathological tissues, that is: mastopathy and carcinoma.

8. Coil-20-product Dataset [108]: Columbia Object Image Library 20 (Coil-20-product) is an object recognition benchmark dataset consisting of grayscaling images from 20 objects. These objects have diversified reflection properties and complex geometric. Each object was rotated 360 degrees by the turntable, and 72 images were taken per object (rotated once every 5 degrees). The object is clipped out from the black background using a rectangular bounding box. The bounding box is resized to 128×128 using interpolation-decimation filters to minimize aliasing.

9. UMIST [152] The face recognition benchmark dataset, Sheffield Face Database(UMIST), consists of 575 images of 20 individuals (mixed race/gender/appearance). Each individual is shown in a range of poses from profile to frontal views.Each subject is with the different view. Each image is re-scaled to 28×23 (https://www.sheffield.ac.uk/eee/research/iel/research/face).

10. ULC [106, 107]: The urban land cover (ULC) data are used to classify high-resolution aerial images which consists of 9 types with 273 training samples, 77 test samples and 147 attributes.

11. MNIST [169]: This handwriting digit data contain 0 to 9 digits images of size 28 by 28 pixels. Each image is reshaped to a 784-dimensional vector for two provirus work. The training set includes $1,000$ samples of each digit, and the test set also consists of $1,000$ of each digit (http://yann.lecun.com/exdb/mnist/).

Here is the synthetic dataset list adopted in this dissertation:

1. Synthetic Data with 4 Features: The synthetic dataset was generated by modifying the dataset used in Griffiths [72]. Specifically, our dataset comprises 6×6 grey images adapted via three different luminance levels, as illustrated in Fig. 4.6(a). Each row of the observations $\mathbf{Y}$ is generated using $\mathbf{Z}$ to linearly combine a subset of four binary factors $\mathbf{X}$ (see Fig. 4.3(a)). In addition, $\mathbf{W}$ loads different luminance combinations. The modified dataset presents a more challenging problem and appears more appropriate for evaluating the different methods.

2. Com-USPS & Pre-USPS Data: We generated two datasets from USPS: the Com-USPS and the Pre-USPS. The digits, used in these two datasets, are sampled randomly from the USPS. Moreover, our generated datasets are scaled to $[0, 1]$. Each row of the Com-USPS dataset is built up with 32×32 grey images. Various kinds of digits $0, 1, 2, 3$ are combined with each sample, as illustrated in Fig. 4.7(a). The Pre-USPS dataset contains merely a single handwritten digit, which is also chosen randomly from $0, 1, 2, 3$. In the training set, each digit has 100 samples. Furthermore, in order to see if the various methods can restore an image, the test samples are bottom-halved from the original images (see Fig. 4.10(b)).

3. Com-NIST & Pre-NIST Data: The rest two datasets were both generated from NIST handprinted forms and characters database. It is worth mentioning that all the images are binary in this database. In the way same as generating Com-USPS, we generated the Com-NIST dataset so that each sample of the Com-NIST consists of 64×64 binary images combined from letters a, b, c, d (see Fig. 4.5(a)). On the other hand, in the Pre-NIST dataset, each sample contains a single handwritten letter chosen randomly from a, c, d. In the training set, each letter has 200 samples. Similarly, the test samples are top-halved from the original images so as to validate if the various methods can restore them (see Fig. 4.11(b)).

4. ULC-3 Dataset [106, 107]: The Urban land cover (ULC) dataset contains 9 types of urban land cover from high-resolution aerial imagery. In this dissertation, for simplicity, we only extract three types of experimental data, that is building, concrete, and grass. This dataset consists of 273 training samples, 77 test samples and 147 attributes.

5. Coil-4-proc Dataset [108]: This dataset contains images for 4 objects discarding the background and each object has 72 samples. The images are downsampled into 32 by 32 pixels and then reshaped to a 1024-dimensional vector. There are just 248 samples in the training set and 40 samples in the test set.

6. Leuk72_3k Dataset [167]: This dataset is an artificial dataset including 3 classes which have been drawn from randomly generated Gaussian mixtures. The Leuk72_3k has only 54 training samples and 18 test samples with 39 attributes.

7. USPS1-4 Dataset [182]: This handwriting digit data contains 1 to 4 digits images of size 16 by 16 pixels. Each image is reshaped to a 256-dimensional vector. The training set includes 100 samples of each digit and the test set also consists of 100 of each digit.

8. CASIA-HWDB: The off-line Chinese handwriting character dataset contains $3,755$ classes of size 64×64 images. The training set includes around 236 samples of each Chinese character and the test set also consists of around 60 of each Chinese character [178]. To facilitate the 2L-MJFA model, we used a benchmark feature extraction method 8-direction histogram feature extraction combined with pseudo-$2D$ bi-moment normalisation for representing a character sample. The feature dimensionality is further reduced to 160 by FDA [179].

   - HZ-20: The extracted data of the first 20 classes.
   - HZ-297: The extracted data of the first 297 classes.

The synthetic datasets are available online at: https://github.com/zzy8989/Data-iNBMT

Reference

[1] C. Robert, *Machine Learning, a Probabilistic Perspective*. Taylor & Francis, 2014.

[2] O. Rippel and R. P. Adams, “High-dimensional probability estimation with deep density models,” *arXiv preprint arXiv:1302.5125*, 2013.

[3] Z. Ghahramani, “Probabilistic machine learning and artificial intelligence,” *Nature*, vol. 521, no. 7553, p. 452, 2015.

[4] C. M. Bishop, “Latent variable models,” in *Learning in Graphical Models*. Springer, 1998, pp. 371–403.

[5] B. D. Haig, *Aspects of Latent Variable Theory*. Taylor & Francis, 2008.

[6] K. E. Masyn, C. E. Henderson, and P. E. Greenbaum, “Exploring the latent structures of psychological constructs in social development using the dimensional–categorical spectrum,” *Social Development*, vol. 19, no. 3, pp. 470–493, 2010.

[7] M. Kadar, “Enterprise interoperability maturity levels assessed through latent trait models,” in *Proceedings of International Conference on Software Engineering, Knowledge Engineering and Information Engineering*, 2015, pp. 75–79.

[8] K. Huang, H. Yang, I. King, and M. R. Lyu, *Machine Learning: Modeling Data Locally and Gloablly*. Springer Verlag, 2008.

[9] X. Yang, K. Huang, R. Zhang, and A. Hussain, “Learning latent features with infinite non-negative binary matrix tri-factorization,” in *Proceedings of International Conference on Neural Information Processing*, 2016, pp. 587–596.

[10] S. R. Waterhouse, D. MacKay, and A. J. Robinson, “Bayesian methods for mixtures of experts,” in *Proceedings of Advances in Neural Information Processing Systems*, 1996, pp. 351–357.

[11] C. Ding, T. Li, W. Peng, and H. Park, "Orthogonal nonnegative matrix tri-factorizations for clustering," in *Proceedings of International Conference on Knowledge Discovery and Data Mining*, 2006, pp. 126–135.

[12] G. E. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[13] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[14] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted boltzmann machines for collaborative filtering," in *Proceedings of International Conference on Machine Learning*, 2007, pp. 791–798.

[15] Z. Ghahramani, "Bayesian non-parametrics and the probabilistic approach to modelling," *Philosophical Transactions of the Royal Society A*, vol. 371, no. 1984, 2013.

[16] X. Yang, K. Huang, and R. Zhang, "Unsupervised dimensionality reduction for gaussian mixture model," in *Proceedings of International Conference on Neural Information Processing*, 2014, pp. 84–92.

[17] X. Yang, K. Huang, R. Zhang, and J. Y. Goulermas, "Two-layer mixture of factor analyzers with joint factor loading," in *Proceedings of International Joint Conference on Neural Networks*, 2015, pp. 1–8.

[18] X. Yang, K. Huang, J. Y. Goulermas, and R. Zhang, "Joint learning of unsupervised dimensionality reduction and gaussian mixture model," *Neural Processing Letters*, vol. 45, no. 3, pp. 791–806, 2017.

[19] X. Yang, K. Huang, R. Zhang, J. Y. Goulermas, and A. Hussain, "A new two-layer mixture of factor analyzers with joint factor loading model for the classification of small dataset problems," *Neurocomputing*, vol. 312, pp. 352–363, 2018.

[20] C. Reed and Z. Ghahramani, "Scaling the indian buffet process via submodular maximization," in *Proceedings of International Conference on Machine Learning*, 2013, pp. 1013–1021.

[21] F. Doshi-Velez and Z. Ghahramani, "Correlated non-parametric latent feature models," in *Proceedings of Conference on Uncertainty in Artificial Intelligence*, 2009, pp. 143–150.

[22] X. Yang, K. Huang, R. Zhang, and A. Hussain, “Learning latent features with infinite nonnegative binary matrix trifactorization,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 3, 2018.

[23] X. Yang, K. Huang, and R. Zhang, “Deep mixtures of factor analyzers with common loadings: A novel deep generative approach to clustering,” in *Proceedings of International Conference on Neural Information Processing*, 2017, pp. 709–719.

[24] X. Yang, K. Huang, R. Zhang, and J. Y. Goulermas, “A novel deep density model for unsupervised learning,” *Cognitive Computation*, pp. 1–11, 2018.

[25] K. Dehnad, *Density Estimation for Statistics and Data Analysis*. Taylor & Francis Group, 1987.

[26] R. P. Duin and D. Tax, “Statistical pattern recognition,” in *Handbook of Pattern Recognition and Computer Vision*. World Scientific, 2005, pp. 3–24.

[27] M. E. Tipping and C. M. Bishop, “Probabilistic principal component analysis,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 61, no. 3, pp. 611–622, 1999.

[28] B. S. Everitt, “Factor analysis,” in *An Introduction to Latent Variable Models*. Dordrecht: Springer Netherlands, 1984, pp. 13–31.

[29] G. E. Hinton, “Deep belief networks,” *Scholarpedia*, vol. 4, no. 5, p. 5947, 2009.

[30] D. Reynolds, “Gaussian mixture models,” *Encyclopedia of Biometrics*, pp. 827–832, 2015.

[31] G. J. McLachlan and D. Peel, “Mixtures of factor analyzers,” in *Proceedings of International Conference on Machine Learning*, 2000, pp. 599–606.

[32] J. K. Vermunt and J. Magidson, “Latent class analysis,” *The Sage Encyclopedia of Social Sciences Research Methods*, pp. 549–553, 2004.

[33] M. C. Neale, S. M. Boker, G. Xie, and H. M. Maes, *Mx: Statistical Modeling*. Richmond, VA: Department of Psychiatry, Virginia Commonwealth University, 1999.

[34] J. Galbraith, I. Moustaki, D. J. Bartholomew, and F. Steele, *The Analysis and Interpretation of Multivariate Data for Social Scientists*. Chapman & Hall/CRC Press, 2002.

[35] B. Everett, *An Introduction to Latent Variable Models.* Springer Science & Business Media, 2013.

[36] J. C. Loehlin, *Latent Variable Models: An Introduction to Factor, Path, and Structural Analysis.* Lawrence Erlbaum Associates Publishers, 1998.

[37] P. Smaragdis, B. Raj, and M. Shashanka, "A probabilistic latent variable model for acoustic modeling," *Advances in Models for Acoustic Processing, NIPS*, vol. 148, pp. 8–1, 2006.

[38] S. Lipovetsky, *Latent Variable Models and Factor Analysis.* Taylor & Francis, 2001.

[39] D. J. Bartholomew, M. Knott, and I. Moustaki, *Latent Variable Models and Factor Analysis: A Unified Approach.* John Wiley & Sons, 2011.

[40] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 1–38, 1977.

[41] A. Mackiewicz and W. Ratajczak, "Principal components analysis (pca)," *Computers and Geosciences*, vol. 19, pp. 303–342, 1993.

[42] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[43] M. E. Tipping and C. M. Bishop, "Mixtures of probabilistic principal component analyzers," *Neural Computation*, vol. 11, no. 2, pp. 443–482, 1999.

[44] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent Component Analysis.* New York: John Wiley & Sons, 2001.

[45] J. Ye, R. Janardan, and Q. Li, "Two-dimensional linear discriminant analysis," in *Advances in neural information processing systems*, 2005, pp. 1569–1576.

[46] I. Jolliffe, *Principal Component Analysis.* Springer Verlag, 1986.

[47] P. Demartines and J. Hérault, "Curvilinear component analysis: A self-organizing neural network for nonlinear mapping of data sets," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 148–154, 1997.

[48] J.-M. Lee, C. Yoo, S. W. Choi, P. A. Vanrolleghem, and I.-B. Lee, "Nonlinear process monitoring using kernel principal component analysis," *Chemical Engineering Science*, vol. 59, no. 1, pp. 223–234, 2004.

[49] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," in *Proceedings of International Conference on Pattern Recognition*, vol. 2, 2004, pp. 28–31.

[50] Z. Ghahramani and G. Hinton, "The em algorithm for mixtures of factor analyzers," in *Technical Report CRG-TR-96-1, University of Toronto*. http://www.gatsby.ucl.ac.uk/.zoubin/papers.html, 1996, pp. 11–18.

[51] Y. Huang, K. B. Englehart, B. Hudgins, and A. D. Chan, "A gaussian mixture model based classification scheme for myoelectric control of powered upper limb prostheses," *IEEE Transactions on Biomedical Engineering*, vol. 52, no. 11, pp. 1801–1811, 2005.

[52] J. A. Bilmes *et al.*, "A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models," *International Computer Science Institute*, vol. 4, no. 510, p. 126, 1998.

[53] A. Montanari and C. Viroli, "Maximum likelihood estimation of mixtures of factor analyzers," *Computational Statistics & Data Analysis*, vol. 55, no. 9, pp. 2712–2723, 2011.

[54] M. A. T. Figueiredo and A. K. Jain, "Unsupervised learning of finite mixture models," *IEEE Transactions on pPattern Analysis and Machine Intelligence*, vol. 24, no. 3, pp. 381–396, 2002.

[55] E. Fokoué, "Mixtures of factor analyzers: an extension with covariates," *Journal of Multivariate Analysis*, vol. 95, no. 2, pp. 370–384, 2005.

[56] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Proceedings of Conference on Computer Vision and Pattern Recognition*. IEEE, 1999, p. 2246.

[57] W. R. Gilks, S. Richardson, and D. Spiegelhalter, *Markov Chain Monte Carlo in Practice*. CRC press, 1995.

[58] E. Helfand, "Theory of inhomogeneous polymers: Fundamentals of the gaussian random-walk model," *The Journal of Chemical Physics*, vol. 62, no. 3, pp. 999–1005, 1975.

[59] J. Ma and L. Xu, "Asymptotic convergence properties of the em algorithm with respect to the overlap in the mixture," *Neurocomputing*, vol. 68, pp. 105–129, 2005.

[60] C. B. Do and S. Batzoglou, "What is the expectation maximization algorithm?" *Nature Biotechnology*, vol. 26, no. 8, pp. 897–899, 2008.

[61] G. McLachlan and T. Krishnan, *The EM Algorithm and Extensions*. John Wiley & Sons, 2007, vol. 382.

[62] C. Bishop, *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2006.

[63] A. Y. Lo, "On a class of bayesian nonparametric estimates: I. density estimates," *The Annals of Statistics*, pp. 351–357, 1984.

[64] I. Porteous, E. Bart, and M. Welling, "Multi-hdp: A non parametric bayesian model for tensor factorization." in *Proceedings of Association for the Advancement of Artificial Intelligence*, vol. 8, 2008, pp. 1487–1490.

[65] P. Müller and F. A. Quintana, "Nonparametric bayesian data analysis," *Statistical Science*, pp. 95–110, 2004.

[66] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei, "Sharing clusters among related groups: Hierarchical dirichlet processes," in *Proceedings of Advances in Neural Information Processing Systems*, 2005, pp. 1385–1392.

[67] Z. Ghahramani, T. L. Griffiths, and P. Sollich, "Bayesian nonparametric latent feature models," *Bayesian Statistics*, vol. 8, pp. 1–25, 2007.

[68] P. Orbanz and J. M. Buhmann, "Nonparametric bayesian image segmentation," *International Journal of Computer Vision*, vol. 77, no. 1-3, pp. 25–45, 2008.

[69] S. J. Gershman and D. M. Blei, "A tutorial on bayesian nonparametric models," *Journal of Mathematical Psychology*, vol. 56, no. 1, pp. 1–12, 2012.

[70] C. E. Rasmussen, "Gaussian processes in machine learning," in *Advanced Lectures on Machine Learning*. Springer, 2004, pp. 63–71.

[71] D. B. Dahl, "Model-based clustering for expression data via a dirichlet process mixture model," *Bayesian Inference for Gene Expression and Proteomics*, vol. 201, p. 218, 2006.

[72] Z. Ghahramani and T. L. Griffiths, "Infinite latent feature models and the indian buffet process," in *Proceedings of Advances in Neural Information Processing Systems*, 2006, pp. 475–482.

[73] R. Thibaux and M. I. Jordan, “Hierarchical beta processes and the indian buffet process,” in *Artificial Intelligence and Statistics*, 2007, pp. 564–571.

[74] C. W. Fox and S. J. Roberts, “A tutorial on variational bayesian inference,” *Artificial intelligence review*, vol. 38, no. 2, pp. 85–95, 2012.

[75] Y. Tang, R. Salakhutdinov, and G. E. Hinton, “Deep mixtures of factor analysers,” in *Proceedings of International Conference on Machine Learning*, 2012.

[76] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, “Greedy layer-wise training of deep networks,” in *Proceedings of Advances in Neural Information Processing Systems*, 2007, pp. 153–160.

[77] L. Arnold and Y. Ollivier, “Layer-wise learning of deep generative models,” *CoRR*, vol. abs/1212.1524, 2012. [Online]. Available: http://arxiv.org/abs/1212.1524

[78] Y. Bengio, “Learning deep architectures for AI,” *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[79] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[80] N. Dilokthanakul, P. A. Mediano, M. Garnelo, M. C. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan, “Deep unsupervised clustering with gaussian mixture variational autoencoders,” *arXiv preprint arXiv:1611.02648*, 2016.

[81] X. Chen, D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel, “Variational lossy autoencoder,” *arXiv preprint arXiv:1611.02731*, 2016.

[82] M. E. Abbasnejad, A. Dick, and A. van den Hengel, “Infinite variational autoencoder for semi-supervised learning,” in *Proceedings of Conference on Computer Vision and Pattern Recognition*, 2017, pp. 781–790.

[83] J. Xie, R. Girshick, and A. Farhadi, “Unsupervised deep embedding for clustering analysis,” in *Proceedings of International Conference on Machine Learning*, 2016, pp. 478–487.

[84] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *arXiv preprint arXiv:1312.6114*, 2013.

[85] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," in *Proceedings of International Conference on Learning Representations*, 2018.

[86] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," in *Proceedings of International Conference on Learning Representations*, 2018.

[87] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards k-means-friendly spaces: Simultaneous deep learning and clustering," *arXiv preprint arXiv:1610.04794*, 2016.

[88] S. Zhai, Y. Cheng, W. Lu, and Z. Zhang, "Deep structured energy based models for anomaly detection," *arXiv preprint arXiv:1605.07717*, 2016.

[89] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proceedings of Conference on Computer Vision and Pattern Recognition*, 2015, pp. 427–436.

[90] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," *arXiv preprint arXiv:1610.02136*, 2016.

[91] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[92] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," *arXiv preprint*, 2017.

[93] T. Devries and G. W. Taylor, "Learning confidence for out-of-distribution detection in neural networks," *arXiv preprint*, 2018.

[94] B. Xu, K. Huang, and C.-L. Liu, "Maxi-min discriminant analysis via online learning," *Neural Networks*, vol. 34, pp. 56–64, 2012.

[95] R. T. Trevor Hastie, "Discriminant analysis by gaussian mixtures," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 155–176, 1996.

[96] K. Huang, I. King, and M. R. Lyu, "Direct zero-norm optimization for feature selection," in *Proceedings of International Conference on Data Mining*, 2008.

[97] K. Huang, D. Zheng, J. Sun, Y. Hotta, K. Fujimoto, and S. Naoi, “Sparse learning for support vector classification,” *Pattern Recognition Letters*, vol. 31, no. 13, pp. 1944–1951, 2010.

[98] K. Huang, H. Yang, M. R. Lyu, and I. King, “Maxi-min margin machine: Learning large margin classifiers localy and globally,” *IEEE Transactions on Neural Networks*, vol. 19, no. 2, pp. 260–272, 2008.

[99] G. J.Mclanchlan, D. Peel, and R. W. Bean, “Modelling high-dimensional data by mixtures of factor analyzers,” *Computational Statistics & Data Analysis*, vol. 41, pp. 379–388, 2003.

[100] J. Baek, G. J. McLachlan, and L. K. Flack, “Mixtures of factor analyzers with common factor loadings: Applications to the clustering and visualization of high-dimensional data,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 7, pp. 1298–1309, 2010.

[101] K. Huang, I. King, and M. R. Lyu, “Finite mixture model of bound semi-naive bayesian network classifier,” in *Proceedings of International Conference on Artificial Neural Networks*, vol. 2714, 2003, pp. 115–122.

[102] ——, “Discriminative training of bayesian chow-liu tree multinet classifiers,” in *Proceedings of International Joint Conference on Neural Network*, vol. 1, 2003, pp. 484–488.

[103] A. Asuncion and D. Newman, “UCI machine learning repository,” in *http://www.ics.uci.edu/~mlearn/MLRepository.html*, 2007.

[104] G.Schwarz, “Estimating the dimension of a model,” *Annals of Statistics*, vol. 6, pp. 461–464, 1978.

[105] L. Hubert and P. Arabie, “Comparing partitions,” *Classification*, vol. 2, pp. 193–218, 1985.

[106] B. Johnson and Z. Xie, “Classifying a high resolution image of an urban area using super-object information,” *Journal of Photogrammetry and Remote Sensing*, vol. 83, pp. 40–49, 2013.

[107] B. Johnson, “High resolution urban land cover classification using a competitive multi-scale object-based approach,” *Remote Sensing Letters*, vol. 4, no. 2, pp. 131–140, 2013.

[108] S. A. Nene, S. K. Nayar, and H. Murase, "Columbia object image library (coil-20)," Technical Report CUCS-005-96, Tech. Rep., February 1996.

[109] M. Lavine and M. West, "A bayesian method for classification and discrimination," *Canadian Journal of Statistics*, vol. 20, no. 4, pp. 451–461, 1992.

[110] Y. Zhang, C. Bingham, M. Gallimore, and J. Chen, "Steady-state and transient operation discrimination by variational bayesian gaussian mixture models," in *Proceedings of International Workshop on Machine Learning for Signal Processing*, 2013, pp. 1–5.

[111] K. Copsey and A. Webb, "Bayesian approach to mixture models for discrimination," in *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, 2000, pp. 491–500.

[112] S. J. Raudys and A. K. Jain, "Small sample size effects in statistical pattern recognition: Recommendations for practitioners," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13(3), pp. 252–264, 1991.

[113] R. Huang, Q. Liu, H. Lu, and S. Ma, "Solving the small sample size problem of LDA," in *Proceedings of International Conference on Pattern Recognition*, 2002, pp. 29–32.

[114] X. Wei and C. Li, "Bayesian mixtures of common factor analyzers: Model, variational inference, and applications," *Signal Processing*, vol. 93, no. 11, pp. 2894–2905, 2013.

[115] W. Wang, "Mixtures of common factor analyzers for high-dimensional data with missing information," *J. Multivariate Analysis*, vol. 117, pp. 120–133, 2013.

[116] G. Hinton, P. Dayan, and M. Revow, "Modeling the manifolds of images of handwritten digits," *IEEE Transactions on Neural Networks*, vol. 8, pp. 65–74, 1997.

[117] A. Basilevsky, *Statistical Factor Analysis and Related Methods*. New York: Wiley, 1994.

[118] M. Kearns, Y. Mansour, and A. Y. Ng, "An information-theoretic analysis of hard and soft assignment methods for clustering," in *Learning in Graphical Models*. Springer, 1998, pp. 495–520.

[119] H. T. Kahraman, S. Sagiroglu, and I. Colak, “The development of intuitive knowledge classifier and the modeling of domain dependent data,” *Knowledge-Based Systems*, vol. 37, no. 13, pp. 283–295, 2013.

[120] W. Street, W. Wolberg, and O. Mangasarian, “Nuclear feature extraction for breast tumor diagnosis,” *IST/SPIE 1993 International Symposium on Electronic Imaging: Science and Technology*, vol. 1905, pp. 861–870, 1993.

[121] O. Mangasarian, W. Street, and W. Wolberg, “Breast cancer diagnosis and prognosis via linear programming,” *Operations Research*, vol. 43(4), pp. 26–30, 1995.

[122] A. Tsanas, M. Little, C. Fox, and L. Ramig, “Objective automatic assessment of rehabilitative speech treatment in parkinson disease,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 22, pp. 181–190, 2014.

[123] J. E. Silva, J. P. Marques de Sa, and J. Jossinet, “Classification of breast tissue by electrical impedance spectroscopy,” *Medical and Biological Engineering and Computing*, vol. 38, pp. 26–30, 2000.

[124] D. D. Lee and H. S. Seung, “Algorithms for non-negative matrix factorization,” in *Proceedings of Advances in Neural Information Processing Systems*, 2001, pp. 556–562.

[125] ——, “Learning the parts of objects by non-negative matrix factorization,” *Nature*, vol. 401, p. 788791, 1999.

[126] T. Li and C. H. Q. Ding, “Nonnegative matrix factorizations for clustering: A survey,” in *Proceedings of Data Clustering: Algorithms and Applications*, 2013, pp. 149–176.

[127] J. Yoo and S. Choi, “Orthogonal nonnegative matrix tri-factorization for co-clustering: Multiplicative updates on stiefel manifolds,” *Information Processing & Management*, vol. 46, no. 5, pp. 559–570, 2010.

[128] R. G. Soares, H. Chen, and X. Yao, “A cluster-based semisupervised ensemble for multiclass classification,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 1, no. 6, pp. 408–420, 2017.

[129] X. Luo, M. Zhou, Y. Xia, and Q. Zhu, “An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems,” *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1273–1284, 2014.

[130] Z. Zhang, T. Li, C. H. Q. Ding, X. Ren, and X. Zhang, "Binary matrix factorization for analyzing gene expression data," *Data Mining and Knowledge Discovery*, vol. 20, no. 1, pp. 28–52, 2010.

[131] H. Wang, F. Nie, H. Huang, and F. Makedon, "Fast nonnegative matrix tri-factorization for large-scale data co-clustering," in *Proceedings of International Joint Conference on Artificial Intelligence*, 2011, pp. 1553–1558.

[132] S. Wang and A. Huang, "Penalized nonnegative matrix tri-factorization for co-clustering," *Expert Systems with Applications*, vol. 78, pp. 64–73, 2017.

[133] B. Fan, Q. Kong, T. Trzcinski, Z. Wang, C. Pan, and P. Fua, "Receptive fields selection for binary feature description," *IEEE Transactions on Image Processing*, vol. 23, no. 6, pp. 2583–2595, 2014.

[134] B. Fan, Q. Kong, W. Sui, Z. Wang, X. Wang, S. Xiang, C. Pan, and P. Fua, "Do we need binary features for 3d reconstruction?" in *Proceedings of Conference on Computer Vision and Pattern Recognition Workshops*, 2016, pp. 53–62.

[135] F. Doshi-velez, K. T. Miller, J. V. Gael, and Y. W. Teh, "Variational inference for the indian buffet process," in *Proceedings of International Conference on Artificial Intelligence and Statistics*, 2009, pp. 137–144.

[136] S. Hasler, H. Wersing, and E. Körner, "Combining reconstruction and discrimination with class-specific sparse coding," *Neural Computation*, vol. 19, no. 7, pp. 1897–1918, 2007.

[137] K. T. Miller, T. L. Griffiths, and M. I. Jordan, "The phylogenetic indian buffet process: A non-exchangeable nonparametric prior for latent features," in *Proceedings of Conference in Uncertainty in Artificial Intelligence*, 2008, pp. 403–410.

[138] D. A. Knowles and Z. Ghahramani, "Infinite sparse factor analysis and infinite independent components analysis," *Independent Component Analysis and Signal Separation*, pp. 381–388, 2007.

[139] R. KRAUSE and D. L. WILD, "Identifying protein complexes in high-throughput protein interaction screens using an infinite latent feature model," in *Proceedings of Advances in Pacific Symposium on Biocomputing*, vol. 11, 2006, pp. 231–242.

[140] D. J. Navarro and T. L. Griffiths, "Latent features in similarity judgments: A non-parametric bayesian approach," *Neural Computation*, vol. 20, no. 11, pp. 2597–2628, 2008.

[141] D. Görür, F. Jäkel, and C. E. Rasmussen, "A choice model with infinitely many latent features," in *Proceedings of International Conference on Machine Learning*, 2006, pp. 361–368.

[142] K. Miller, M. I. Jordan, and T. L. Griffiths, "Nonparametric latent feature models for link prediction," in *Proceedings of Advances in Neural Information Processing Systems 22*, 2009, pp. 1276–1284.

[143] H. P. Dang and P. Chainais, "Indian buffet process dictionary learning: Algorithms and applications to image processing," *International Journal of Approximate Reasoning*, vol. 83, pp. 1–20, 2017.

[144] Q. Pan, D. Kong, C. H. Q. Ding, and B. Luo, "Robust non-negative dictionary learning," in *Proceedings of Conference on Artificial Intelligence*, 2014, pp. 2027–2033.

[145] Z. Ghahramani and M. J. Beal, "Variational inference for bayesian mixtures of factor analysers," in *Proceedings of Advances in Neural Information Processing Systems*, 1999, pp. 449–455.

[146] H. Attias, "A variational baysian framework for graphical models," in *Proceedings of Advances in Neural Information Processing Systems*, 1999, pp. 209–215.

[147] Z. Ghahramani and M. J. Beal, "Propagation algorithms for variational bayesian learning," in *Proceedings of Advances in Neural Information Processing Systems*, 2000, pp. 507–513.

[148] S. Gershman, P. I. Frazier, and D. M. Blei, "Distance dependent infinite latent feature models," *EEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 2, pp. 334–345, 2015.

[149] T. L. Griffiths and Z. Ghahramani, "The indian buffet process: An introduction and review," *Journal of Machine Learning Research*, vol. 12, pp. 1185–1224, 2011.

[150] P. Rai and H. D. III, "Multi-label prediction via sparse infinite CCA," in *Proceedings of Advances in Neural Information Processing Systems*, 2009, pp. 1518–1526.

[151] S. Williamson, P. Orbanz, and Z. Ghahramani, "Dependent indian buffet processes," in *Proceedings of International Conference on Artificial Intelligence and Statistics*, 2010, pp. 924–931.

[152] D. B. Graham and N. M. Allinson, "Characterising virtual eigensignatures for general purpose face recognition," in *Face Recognition*. Springer, 1998, pp. 446–456.

[153] I. S. Dhillon and J. A. Tropp, "Matrix nearness problems with bregman divergences," *SIAM J. Matrix Analysis Applications*, vol. 29, no. 4, pp. 1120–1146, 2007.

[154] B. Kulis, M. A. Sustik, and I. S. Dhillon, "Low-rank kernel learning with bregman matrix divergences," *Journal of Machine Learning Research*, vol. 10, pp. 341–376, 2009.

[155] P. Yang, K. Huang, and C. Liu, "Geometry preserving multi-task metric learning," *Machine Learning*, vol. 92, no. 1, pp. 133–175, 2013.

[156] P. Yang, K. Huang, and C.-L. Liu, "A multi-task framework for metric learning with common subspace," *Neural Computing and Applications*, vol. 22, no. 7-8, pp. 1337–1347, 2013.

[157] H. Wang, H. Huang, C. Ding, and F. Nie, "Predicting protein–protein interactions from multimodal biological data sources via nonnegative matrix tri-factorization," *Journal of Computational Biology*, vol. 20, no. 4, pp. 344–358, 2013.

[158] C. Li, Z.-Y. Liu, X. Yang, Jianhua-Su, and H. Qiao, "Stitching contaminated images," *Neurocomputing*, vol. 214, pp. 829–836, 2016.

[159] Z.-Y. Liu and H. Qiao, "Gnccp - graduated nonconvexity and concavity procedure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36(6), pp. 1258–1267, 2014.

[160] X. Xu, F. Shen, Y. Yang, D. Zhang, H. T. Shen, and J. Song, "Matrix tri-factorization with manifold regularizations for zero-shot learning," in *Proceeding of Conference on Computer Vision and Pattern Recognition*, 2017.

[161] A. B. Patel, T. Nguyen, and R. G. Baraniuk, "A probabilistic theory of deep learning," *arXiv preprint arXiv:1504.00641*, 2015.

[162] R. P. Adams, H. M. Wallach, and Z. Ghahramani, "Learning the structure of deep sparse graphical models," in *Proceedings of International Conference on Artificial Intelligence and Statistics*, 2010, pp. 1–8.

[163] J. Zhang, S. Ding, N. Zhang, and Y. Xue, "Weight uncertainty in boltzmann machine," *Cognitive Computation*, vol. 8, no. 6, pp. 1064–1073, 2016.

[164] J. Baek and G. J. McLachlan, "Mixtures of common *t*-factor analyzers for clustering high-dimensional microarray data," *Bioinformatics*, vol. 27, no. 9, pp. 1269–1276, 2011.

[165] C. Tortora, P. D. McNicholas, and R. P. Browne, “A mixture of generalized hyperbolic factor analyzers,” *Adv. Data Analysis and Classification*, vol. 10, no. 4, pp. 423–440, 2016.

[166] S. Y. Kung, M.-W. Mak, and S.-H. Lin, *Biometric authentication: a machine learning approach*. Prentice Hall Professional Technical Reference Upper Saddle River, 2005, ch. Expectation-Maximization Theory.

[167] A. Likas, N. Vlassis, and J. J. Verbeek, “The global k-means clustering algorithm,” *Pattern Recognition*, vol. 36, no. 2, pp. 451 – 461, 2003.

[168] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[169] L. Deng, “The mnist database of handwritten digit images for machine learning research [best of the web],” *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[170] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural Networks*, vol. 61, pp. 85–117, 2015.

[171] C. F. Higham and D. J. Higham, “Deep learning: An introduction for applied mathematicians,” *arXiv preprint arXiv:1801.05894*, 2018.

[172] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*. MIT press Cambridge, 2016, vol. 1.

[173] Z. Ghahramani, “A history of bayesian neural networks,” in *Proceedinds of Advances in Neural Information Processing Systems Workshop on Bayesian Deep Learning*, 2016.

[174] G. Marcus, “Deep learning: A critical appraisal,” *arXiv preprint arXiv:1801.00631*, 2018.

[175] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, “Deepfool: a simple and accurate method to fool deep neural networks,” in *Proceedings of Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2574–2582.

[176] Y. Gal and Z. Ghahramani, “Dropout as a bayesian approximation: Representing model uncertainty in deep learning,” in *Proceedings of International Conference on Machine Learning*, 2016, pp. 1050–1059.

[177] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Proceedings of Advances in Neural Information Processing Systems*, 2014, pp. 3581–3589.

[178] C.-L. Liu, F. Yin, D.-H. Wang, and Q.-F. Wang, "Casia online and offline chinese handwriting databases," in *Proceedings of International Conference on Document Analysis and Recognition*, 2011, pp. 37–41.

[179] C.-L. Liu and X.-D. Zhou, "Online japanese character recognition using trajectory-based normalization and direction feature extraction," in *Proceedings of International Workshop on Frontiers in Handwriting Recognition*, 2006.

[180] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[181] M. Charytanowicz, J. Niewczas, P. Kulczycki, P. A. Kowalski, S. Łukasik, and S. Żak, "Complete gradient clustering algorithm for features analysis of x-ray images," in *Information technologies in biomedicine*. Springer, 2010, pp. 15–24.

[182] J. J. Hull, "A database for handwritten text recognition research," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 16, no. 5, pp. 550–554, 1994.