Cognitive Computation (2019) 11:778–788
https://doi.org/10.1007/s12559-018-9566-9

A Novel Deep Density Model for Unsupervised Learning

Xi Yang[1] · Kaizhu Huang[1] · Rui Zhang[1] · John Y. Goulermas[2]

Received: 2 February 2018 / Accepted: 22 May 2018 / Published online: 25 June 2018

Abstract
Density models are fundamental in machine learning and have received a widespread application in practical cognitive modeling tasks and learning problems. In this work, we introduce a novel deep density model, referred to as deep mixtures of factor analyzers with common loadings (DMCFA), with an efficient greedy layer-wise unsupervised learning algorithm. The model employs a mixture of factor analyzers 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 utilizing Gaussian distributions as the priors. Our model is evaluated on five real datasets and compared to three other competitive models including mixtures of factor analyzers (MFA), MFA with common loadings (MCFA), deep mixtures of factor analyzers (DMFA), and their collapsed counterparts. The results demonstrate the superiority of the proposed model in the tasks of density estimation, clustering, and generation.

Keywords Deep density model · Mixtures of factor analyzers · Common component factor loadings · Dimensionality reduction

Introduction

Density models constitute a family of effective machine learning methodologies that connect density estimation and manifold discovery [8, 22]. They are employed to solve many core tasks that estimate probabilities in terms of the data distributions. In the recent years, they have been receiving increasing interest in multi-layered or deep architectures. Compared to shallow models of similar scale, deep models greatly reduce the computational cost of learning, tend to resist overfitting through parameter sharing between successive layers [24], and can effectively promote cognitive capabilities [5, 7]. The knowledge of cognitive agents can be modeled 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 unsupervised learning, prediction, reconstruction, clustering, and simulation [21]. Furthermore, probabilistic graphical models have always had a fundamental role in constructing sophisticated density estimates in deep density models, such as the restricted Boltzmann machine (RBM), Gaussian restricted Boltzmann machine (GRBM), and directed belief networks (DBNs) [1, 10, 11]. Despite their lower computational cost 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 [23, 24, 32].

To mitigate this issue, Tang et al. proposed a deep density model utilizing a greedy layered unsupervised learning algorithm, referred to as deep mixtures of factor analyzers (DMFA) [24].[1] Unlike previous methods, this model is

✉ Kaizhu Huang
Kaizhu.Huang@xjtlu.edu.cn

Xi Yang
xi.yang@xjtlu.edu.cn

Rui Zhang
rui.zhang02@xjtlu.edu.cn

John Y. Goulermas
j.y.goulermas@liverpool.ac.uk

1 Xi'an Jiaotong-Liverpool University, SIP, Suzhou, 215123, China

2 University of Liverpool, Liverpool, L69 3BX, UK

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

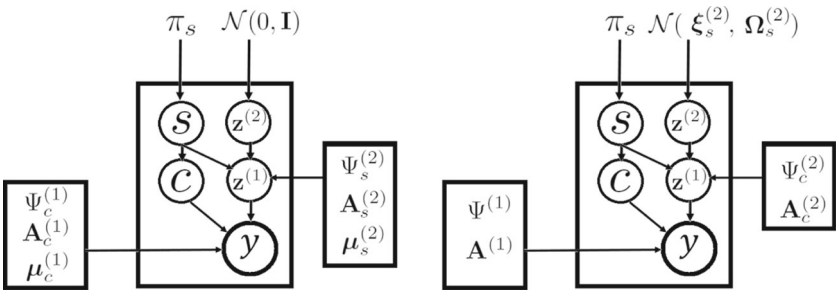
a directed graphical model which has been developed by adopting mixtures of factor analyzers (MFA) [19]. In particular, DMFA extends the same scheme as MFA to train each hidden layer and takes the expectation-maximization (EM) algorithm to maximize the log-likelihood in learning [9, 18]. Its inference and parameter computation procedure is more straightforward than previous methods. However, it is a highly parameterized model where the number of parameters may not be manageable. In real applications, overfitting could become a severe problem in DMFA, because 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.

This work addresses the previous limitations by proposing a novel greedy layer-wise learning approach, referred to as the deep mixtures of factor analyzers with common loadings (DMCFA). In developing the underlying idea, we extend the MFA model sharing a common component factor loading (MCFA) [4] 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 [3, 25]. 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 [26]. The proposed model is also flexible in estimating the data density by utilizing the learnable Gaussian distributions as the priors for each latent unit.

Figure 1b 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 DMFA. Under a two-layer model, the total number of free parameters in DMCFA is far smaller than that in DMFA; approximately, we 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 utilizes approximately only $1/c$ parameters of DMFA (details can be found in Section "Complexity Analysis"). Therefore, our model has the notable advantage of dealing with multivariate data with a larger number of clusters or with insufficient instances.

To optimize 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 two-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 [2]. This makes DMCFA inference and learning more straightforward and efficient and less likely to get trapped into a local optima. Furthermore, the overfitting risk and the computational cost are reduced by sharing the factor loadings among 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.

The remainder of this paper is structured as follows. Section "Related Work" introduces some related work, while the MFA and MCFA approaches are briefly reviewed as background in section "Mixtures of Factor Analyzers with Common Factor Loadings." The proposed method and inference procedure are described in Section "Deep Mixtures of Factor Analyzers with Common Factor Loadings." Section "Experiments" 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 DMCFA over MFA, DMFA, and MCFA, and the shallow forms



(a) Graphical model of DMFA. (b) Graphical model of DMCFA.

Fig. 1 Graphical models of a two-layer DMCFA and DMFA

collapsed from the deep models. Finally, a conclusion is given in the last section. This paper is an extension of [30], based on the reorganization of the method description, supplementing experiments for the generation capability, and expanding the details of related work and the computational complexity.

Related Work

In the recent years, deep learning has developed very rapidly, initiating interest in designing a new type of deep architectures from classical shallow models, such as using the shallow feature learning modules to derive deep architectures [33], or combining simple linear projections, such as PCA, with a CNN learning structure [34]. Deep density models are also frequently used in cognitive computation and deep learning communities. They have been used for several applications, such as feature extraction, object discrimination, quantitative analysis, and medical diagnosis [27, 28, 31, 32]. Recent efforts have moved toward training deep density models with directed graphical models, aiming to find structures within the latent space in an unsupervised manner. For example, [6] focused on learning convolutional factor analysis with non-parametric Bayesian priors. [22] learned a latent representation with a simple factorial density function by constructing bijective deterministic maps. Inspired by the variational auto-encoder (VAE), a deep generative model is used to estimate data density with neural networks, from which unseen samples can be generated [12]. These deep generative models can encode rich latent structures.

The work most related to the proposed model is the DMFA, which is a deep directed graphical model utilizing the multi-layer factor analyzers developed by adopting an MFA in each hidden layer [19, 24]. 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. Figure 1a 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 $T = spq - (cpq + sqd) = q\,(p(s-c) - ds) > 0$.

Mixtures of Factor Analyzers with Common Factor Loadings

The MCFA model considered here is fundamental to the proposed method. For completeness, we briefly review the formulation of MFA models and then introduce the MCFA one. 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 $\boldsymbol{\epsilon}$. In general, q is less than p and use this to analyze high-dimensional patterns. By separating the observations independently into c non-overlapping components, the MFA approach is modeled as

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

where $\mathbf{M}_c \in \mathbb{R}^p$ is the mean vector of each component, $\mathbf{W}_c \in \mathbb{R}^{p \times q}$ is the factor loading matrix of each component, $\mathbf{z}_c$ denotes the $q \times p$ matrix of the c^{th} component factor, and π_c denotes the mixing proportion. The independent noise $\boldsymbol{\epsilon}_c$ follows $\mathcal{N}(0, \Psi_c)$. The latent factor follows a normal distribution with zero mean and an identity covariance matrix $\mathbf{I}_q$. Differently, MCFA assumes that the prior of the latent factor follows a Gaussian density with mean $\boldsymbol{\xi}_c$ and covariance Ω_c, defined via

$$p(\mathbf{z}|c) = \mathcal{N}(\mathbf{z}_c; \boldsymbol{\xi}_c, \Omega_c).$$

By introducing the common factor loadings $\mathbf{A} \in \mathbb{R}^{p\times q}$, the directed generative MCFA model is defined as

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

The vector of independent noise $\boldsymbol{\epsilon}$ follows $\mathcal{N}(0, \Psi)$, where Ψ is a diagonal $p \times p$ matrix, while $c \in 1, \ldots, C$ denotes the component indicator over the C total components of the mixture. π_c denotes the mixing proportions, that is

$$p(c) = \pi_c, \qquad \sum_{c=1}^{C}\pi_c = 1.$$

With the above definitions, the probability density function of the visible variable $\mathbf{y}$, given the latent variables c, and $\mathbf{z}$ can be written as

$$p(\mathbf{y}|c, \mathbf{z}) = \sum_{c=1}^{C}\pi_c \mathcal{N}(\mathbf{y}; \mathbf{A}\mathbf{z}_c, \Psi).$$

This density can also be written through a shallow form by integrating out the latent factors, as

$$\begin{aligned} p(\mathbf{y}|c) &= \int_{\mathbf{z}} p(\mathbf{y}|c, \mathbf{z})p(\mathbf{z}|c)d\mathbf{z} = \mathcal{N}(\mathbf{y}; \boldsymbol{\mu}_c, \Sigma_c), \\ \boldsymbol{\mu}_c &= \mathbf{A}\boldsymbol{\xi}_c, \quad \Sigma_c = \mathbf{A}\Omega_c\mathbf{A}^T + \Psi. \end{aligned} \tag{2}$$

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

$$\begin{aligned} p(\mathbf{y}) &= \sum_{c=1}^{C} p(c)p(\mathbf{y}|c), \\ \mathcal{MCFA}(\mathbf{y}; \boldsymbol{\theta}) &= \sum_{c=1}^{C}\pi_c \mathcal{N}(\mathbf{y}; \mathbf{A}\boldsymbol{\xi}_c, \mathbf{A}\Omega_c\mathbf{A}^T + \Psi). \end{aligned}$$

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

Following the parameter analysis in [29], we can estimate the difference of the number of parameters between MFA and MCFA to be $(c-1)p+[c(q+1)-q](p-q)$, assuming a mixture model setting with $c > 1$ and $p > q$. This result indicates that, compared to MFA, the MCFA model can achieve significant parameter reduction.

Inference

For inference, the posterior probability over the components of the mixture can be found by

$$q(\mathbf{z}, c|\mathbf{y}; \boldsymbol{\theta}) = pr\{\omega_c = 1|\mathbf{y}\}\,.$$

If $\mathbf{y}$ belongs to the c^{th} component, we have $\omega_c = 1$; otherwise, $\omega_c = 0$. $\mathbb{E}[\omega_c|\mathbf{y}]$ denotes the expected value of the component labels. By using Bayes' rule, the formulation of the posterior can be expressed as

$$\begin{aligned} q(c|\mathbf{y}; \boldsymbol{\theta}) &= \frac{p(\mathbf{y}|c)p(c)}{\sum_{h=1}^{C} p(\mathbf{y}|h)p(h)} \\ &= \frac{\pi_c \mathcal{N}(\mathbf{y}; \mathbf{A}\boldsymbol{\xi}_c, \mathbf{A}\Omega_c\mathbf{A}^T + \Psi)}{\sum_{h=1}^{C}\pi_h \mathcal{N}(\mathbf{y}; \mathbf{A}\boldsymbol{\xi}_h, \mathbf{A}\Omega_h\mathbf{A}^T + \Psi)}. \end{aligned} \tag{3}$$

More concisely, the posterior distribution of the latent factors can be given as

$$p(\mathbf{z}|\mathbf{y}, c) = \mathcal{N}(\mathbf{z}; \boldsymbol{\kappa}_c, \mathbf{V}_c^{-1}), \tag{4}$$

$$\mathbf{V}_c^{-1} = \Omega_c^{-1} + \mathbf{A}^T\Psi^{-1}\mathbf{A}, \tag{5}$$

$$\boldsymbol{\kappa}_c = \boldsymbol{\xi}_c + \mathbf{V}_c^{-1}\mathbf{A}^T\Psi^{-1}(\mathbf{y} - \boldsymbol{\mu}_c).$$

This is also a multivariate Gaussian density on $\mathbf{z}$ given $\mathbf{y}$ and c. The log-likelihood function for $\boldsymbol{\theta}$ based on the complete data $\mathbf{y}$ is then given by

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}|\mathbf{y}, \omega, \mathbf{z}) = \sum_{c=1}^{C}\int_{z} q(\mathbf{z}, c|\mathbf{y}; \boldsymbol{\theta}) \\ \{\log p(\mathbf{y}|c, \mathbf{z}; \boldsymbol{\theta}) + \log p(\mathbf{z}|c) + \log \pi_c\} d\mathbf{z}, \end{aligned} \tag{6}$$

where $q(\mathbf{z}, c|\mathbf{y}; \boldsymbol{\theta})$ is the posterior distribution. The bound is tight when $q(\mathbf{z}, c|\mathbf{y}; \boldsymbol{\theta}) = p(\mathbf{z}, c|\mathbf{y}; \boldsymbol{\theta})$.

The parameter estimation of MCFA is conducted with an EM framework. Hence, the E-step requires evaluating the conditional expectation of Eq. 6, which is termed as the Q-function

$$\mathbf{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(k)}) = \mathbb{E}[\mathcal{L}(\boldsymbol{\theta}|\mathbf{y}, \omega, \mathbf{z})],$$

where $\boldsymbol{\theta}^{(k)}$ denotes the current estimate. During the M-step, maximizing the expected log-likelihood is equivalent to having $\partial\mathbf{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(k)})/\partial\boldsymbol{\theta} = 0$ for all parameters and updating them accordingly.

Deep Mixtures of Factor Analyzers with Common Factor Loadings

This section describes how to generalize 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. After a shallow MCFA training reaches convergence, the prior of latent

factors in the first layer will be replaced by an MCFA prior

$$p(\mathbf{z}|c) = \mathcal{MCFA}(\mathbf{z}_c^{(1)}; \boldsymbol{\theta}_c^{(2)}). \tag{7}$$

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 a sample drawn using Eq. 4, and $\boldsymbol{\theta}_c^{(2)}$ denotes the new parameters for the second MCFA layer specific to the component c of the first layer. The same scheme can be extended to train more layers.

In the deep model, $m_c \in \{1, \ldots, M_c\}$ is set to be the sub-component indicator variable,[2] 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

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

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

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

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

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

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

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

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

It is also necessary to consider the following density functions

$$\begin{aligned} p(\mathbf{z}^{(1)}, c|s, \mathbf{z}^{(2)}) &= \mathcal{N}(\mathbf{z}^{(1)}; \mathbf{A}_c^{(2)}\mathbf{z}^{(2)}, \Psi_c^{(2)}), \\ p(\mathbf{z}^{(2)}|s) &= \mathcal{N}(\mathbf{z}^{(2)}; \boldsymbol{\xi}_s^{(2)}, \Omega_s^{(2)}). \end{aligned}$$

According to Eq. 2, the Gaussian density can be expressed on the observed data $\mathbf{y}$ given $\mathbf{z}^{(1)}$ and c, as

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

[2]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.

Here, $\mathbf{A}^{(1)} \in \mathbb{R}^{p\times q}$, $\Psi^{(1)} \in \mathbb{R}^{p\times p}$, $\mathbf{z}^{(1)} \in \mathbb{R}^{q}$, $\mathbf{A}_c^{(2)} \in \mathbb{R}^{q\times d}$, $\Psi_c^{(2)} \in \mathbb{R}^{q\times q}$, $\mathbf{z}^{(2)} \in \mathbb{R}^{d}$, $\boldsymbol{\xi}_s^{(2)} \in \mathbb{R}^{d}$, and $\Omega_s^{(2)} \in \mathbb{R}^{d\times d}$.[3]

The deep model can also be collapsed into a standard MCFA by integrating out the first latent factors. According to Eq. 3, if the first layer factors $\mathbf{z}^{(1)}$ are integrated out, we obtain

$$\begin{aligned} p(\mathbf{y}|\mathbf{z}^{(2)}, s) &= \int_{\mathbf{z}^{(1)}} p(\mathbf{y}|c, \mathbf{z}^{(1)})p(\mathbf{z}^{(1)}|s, \mathbf{z}^{(2)})p(\mathbf{z}^{(2)}|s)d\mathbf{z}^{(1)} \\ &= \mathcal{N}(\mathbf{y}; \mathbf{A}^{(1)}(\mathbf{A}_c^{(2)}\mathbf{z}^{(2)}), \mathbf{A}^{(1)}\Psi_c^{(2)}\mathbf{A}^{(1)^T} + \Psi^{(1)}). \end{aligned}$$

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

$$p(\mathbf{y}|s) = \int_{\mathbf{z}^{(2)}} p(\mathbf{y}|\mathbf{z}^{(2)}, s)d\mathbf{z}^{(2)} = \mathcal{N}(\mathbf{y}; \boldsymbol{\mu}_s, \Sigma_s), \tag{8}$$

$$\begin{aligned} \boldsymbol{\mu}_s &= \mathbf{A}^{(1)}(\mathbf{A}_c^{(2)}\xi_s^{(2)}), \\ \Sigma_s &= \mathbf{A}^{(1)}(\mathbf{A}_c^{(2)}\Omega_s^{(2)}\mathbf{A}_c^{(2)^T} + \Psi_c^{(2)})\mathbf{A}^{(1)^T} + \Psi^{(1)}. \end{aligned} \tag{9}$$

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

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

The parameters of the shallow form of DMCFA can be similarly denoted as $\boldsymbol{\theta}_s = \{\pi_s, \mathbf{A}, \Psi, \mathbf{A}_c, \Psi_c, \xi_s, \Omega_s\}$, for $s = 1, \ldots, S$, and $c = 1, \ldots, C$.

Inference and Learning

According to Eq. 7, we sample $\mathbf{z}^{(1)} \sim p(\mathbf{z}|\mathbf{y}, c)$ by selecting the component $\hat{c} = \underset{c}{\text{argmax}}\ q(c|\mathbf{y}; \boldsymbol{\theta})$. Then $\mathbf{z}^{(1)}$ and $\hat{c}$ are treated as the input data for the second layer. The posterior probability $q(s|\mathbf{z}^{(1)}, \hat{c})$ and the posterior distribution $q(\mathbf{z}^{(2)}, s|\mathbf{z}^{(1)}, \hat{c})$ are computed similarly by the inference for the first layer shown in Eq. 3,4.

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 optimization, 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 in Eq. 6. With regard to the second-layer parameters $\boldsymbol{\theta}_c^{(2)}$ which are specified for component c of the first layer, the DMFA formulation seeks to substitute a more effective prior $\log p(\mathbf{z}^{(1)}|\hat{c}; \boldsymbol{\theta}_c^{(2)})$.

[3]The superscript represents which layer these variables belong 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 $\Psi_c^{(2)}$ are marked with the subscript c. d corresponds to the subspace dimensionality in the second layer, where $d < q$.

Holding the first-layer parameters fixed, maximizing (6) with the second-layer parameters is equivalent to maximizing

$$\mathcal{L}(\boldsymbol{\theta}_c^{(2)}|\mathbf{z}^{(1)}) = \sum_{s\in c}\int_{\mathbf{z}^{(2)}} q(\mathbf{z}^{(2)}, s|\mathbf{z}^{(1)}, \hat{c}; \boldsymbol{\theta}_c^{(2)}) \{\log p(\mathbf{z}^{(1)}, \hat{c}|\mathbf{z}^{(2)}, s; \boldsymbol{\theta}_c^{(2)}) + \log p(\mathbf{z}^{(2)}|s) + \log p(s)\} d\mathbf{z}^{(2)}. \quad (10)$$

The second-layer parameter vector consists of $\boldsymbol{\theta}_c^{(2)} = \{\pi_s, \mathbf{A}_c, \Psi_c, \xi_s, \Omega_s\}$, for $s \in c$, and $c = 1, \ldots, C$.

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 is sampled depending on the posteriors from the first layer. The modified EM algorithm is developed to alter the M-step into two-layer loops. The procedure is summarized in Algorithm 1.

Algorithm 1 The procedure of the 2-loop M-step

Input : Initialized paremeters $\boldsymbol{\theta} = \{\pi_c, \mathbf{A}, \boldsymbol{\xi}_c, \boldsymbol{\Omega}_c, \mathbf{D}\}$, 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 maximization of $\mathbb{E}[\mathcal{L}(\boldsymbol{\theta}_c|\mathbf{y}, \omega_c, \mathbf{z}_c)]$.
  - Calculate partial derivatives of the expectation equations for each global parameters $\partial\mathbf{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(k)})/\partial\boldsymbol{\theta}_g = 0$.
- **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 π_c, $\boldsymbol{\xi}_c$, $\boldsymbol{\Omega}_c$ by calculating partial derivatives of the expectation equations for each local parameters $\partial\mathbf{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(k)})/\partial\boldsymbol{\theta}_l = 0$.

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 Ω_c. Therefore, the updated estimates $\boldsymbol{\xi}_c$ and Ω_c are adjusted by setting $\boldsymbol{\xi}_c \leftarrow \mathbf{U}\boldsymbol{\xi}_c$ and $\Omega_c \leftarrow \mathbf{U}\Omega_c\mathbf{U}^T$.

Every deep density model can be converted to an equivalent shallow form with the same density function. With the comparison of the layer-wise training, the shallow MCFA collapsed from a DMCFA is also possible to run EM steps. This process can be thought of as a "backfitting" procedure beneficial in preventing overfitting. In this case, the density on $\mathbf{y}$ given c can be written as in Eq. 8. For the s^{th} mixture, the posterior probability of the shallow MCFA can be expressed as

$$q(s|\mathbf{y}; \boldsymbol{\theta}_s) = \frac{\pi_s \mathcal{N}(\mathbf{y}; \boldsymbol{\mu}_s, \Sigma_s)}{\sum_{h=1}^{S} \pi_h \mathcal{N}(\mathbf{y}; \boldsymbol{\mu}_h, \Sigma_h)}.$$

We are also interested in the posterior distribution of the latent factor $\mathbf{z}_s$ which is collapsed to a shallow form, described by

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

The DMCFA roughly reduces $(s - c)qd$ parameters from this shallow form. This shows that the shallow learning method can learn the same functions theoretically, but requires a much larger number of parameters than the deep learning methods [5].

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 dimension, c and s are the number 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 [15]. 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 $\frac{q(q-1)}{2}$

Table 1 Parameter settings

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

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

constraints. From all 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.

Experiments

In this section, we evaluate the DMCFA's performance for model-based density estimation and clustering using real datasets with two standard models MFA and MCFA, one deep model DMFA, and the shallow forms collapsed by the deep models. Moreover, we conduct a qualitative experiment on a benchmark dataset to evaluate the performance in terms of generation.

Experimental Setup

In the experiment, we follow the work of DMFA [24], 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 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.

Datasets Description

We use an artificial dataset, a multivariate physical dataset and three 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 three types with 273 training samples, 77 test samples, and 147 attributes which are collected by classifying a high-resolution aerial image [13, 14].
- Leuk72_3k: This is an artificial dataset drawn from randomly generated Gaussian mixtures [17]. There are

Table 2 Performance on various real data in terms of the log-likelihood (the larger, the better) on the training set

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

DMFA and DMCFA are each set to two layers. S-MFA and S-MCFA denote the shallow form by collapsing the deep models. The outperformed results are marked in italics in the table

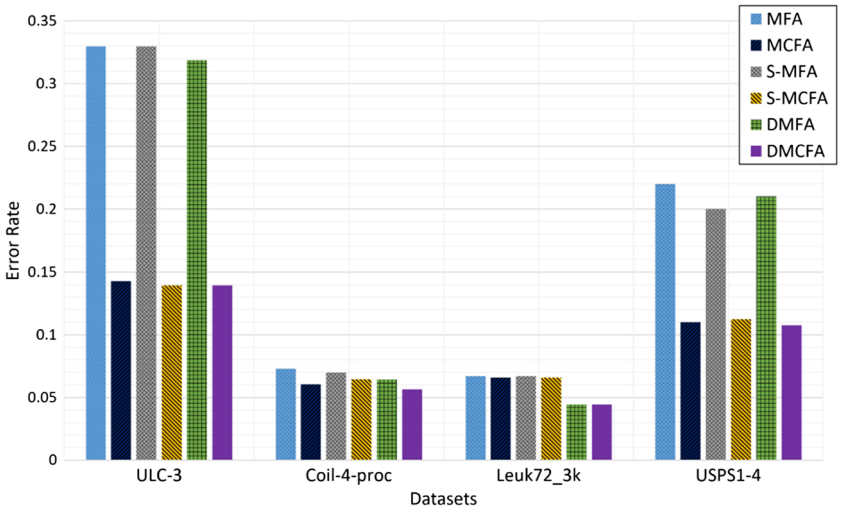


Fig. 2 Clustering performance (error rate) on all the datasets. The best result is reported from each model on training sets

three classes with 39 attributes including 54 training samples and 18 test samples.

- Coil-4-proc: This dataset is a collection of four objects consisting of gray-scale images and each object has 72 samples [20]. These images discard the background 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 random sampling.
- USPS1-4: This dataset contains 16 × 16 images of one to four 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[4] which is composed of 60,000 28 × 28 handwriting digit images [16].

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 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. 2). This may be caused by overfitting due to the limited samples. The mean log-probabilities values on the test set are listed in Table 3 . 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 four real datasets for comparing the performance of DMCFA with the other methods. In the experiments, all of the methods have been initialized by random assortment. The clustering results for the deep models versus the shallow models on the training sets are shown in Fig. 2, and the results on the test sets are shown in Fig. 3. 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 in the test set. Moreover, it can be clearly observed that the collapsed shallow forms 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.

Qualitative Results

To demonstrate the generation results of all models, a qualitative study is conducted on the MNIST dataset. In

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

Table 3 Performance on various real data in terms of the log-likelihood (the larger, the better) on the test set

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

DMFA and DMCFA are each set to two layers. S-MFA and S-MCFA denote the shallow form by collapsing the deep models. The outperformed results are marked in italics in the table

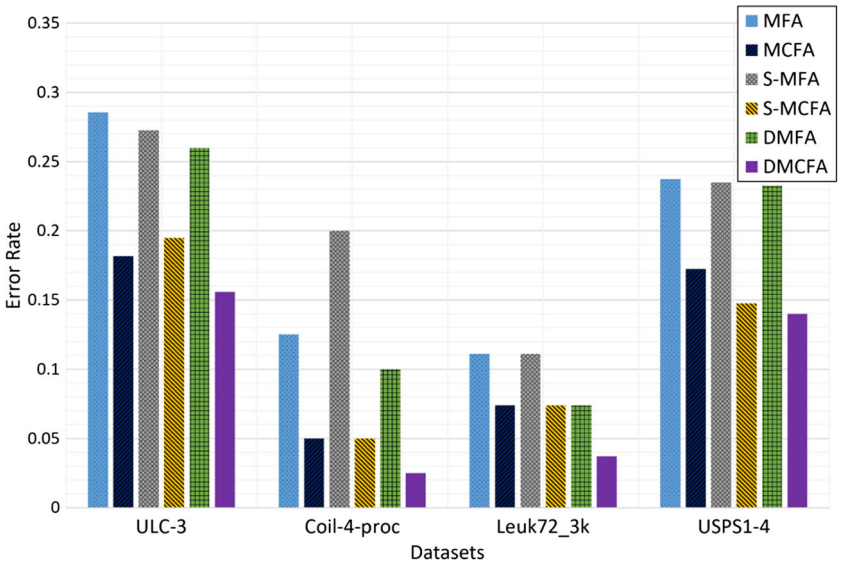
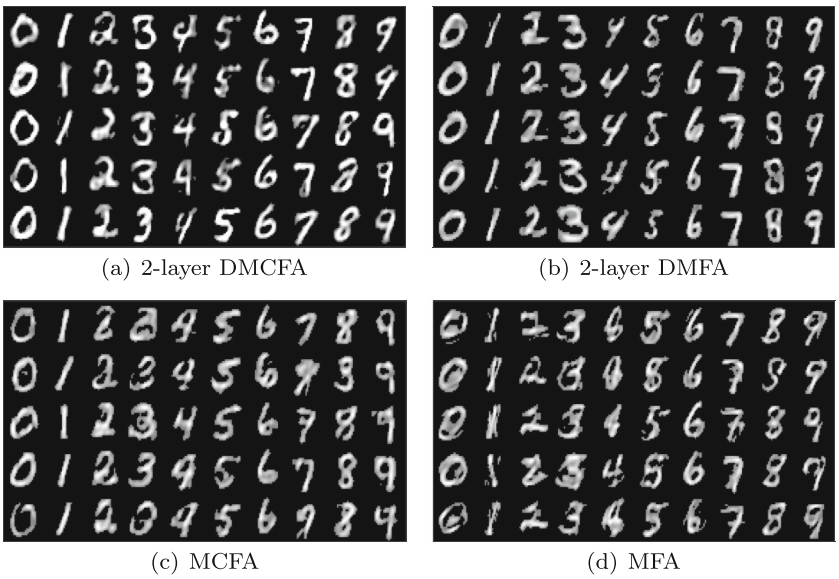


Fig. 3 Clustering performance (error rate) on all the datasets. The best result is reported from each model on test sets



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

(c) MCFA (d) MFA

Fig. 4 Comparison of **a** two-layer DMCFA, **b** two-layer DMFA, **c** MCFA, and **d** MFA on the MNIST dataset for generation

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 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. Figure 4 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. 4a, although the generated 4s and 9s 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. 4b. 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. 4c, d. Obviously, the results of the deep models are complete and more clear.

Conclusion and Future Work

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-maximization algorithm to maximize the posterior and learn the parameters. Compared with existing deep density models, this one enjoys an easy inference procedure, lower time complexity, and a significantly smaller number of free parameters. We evaluate our model on empirical and clustering tasks using real-world datasets, which is shown to achieve better results compared to standard and 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. Future work will include the extension of the idea of common loadings to other deep density models, and also investigation of Bayesian methods, so that the number of mixtures and dimensions of the latent representation space can be automatically learned.

Funding Information The work reported in this paper was partially supported by the following: National Natural Science Foundation of China (NSFC) under grant no. 61473236, Natural Science Fund for Colleges and Universities in Jiangsu Province under grant no. $17KJD520010$, Suzhou Science and Technology Program under grant nos. $SYG201712$ and $SZS201613$, Jiangsu University Natural Science Research Programme under grant no. $17KJB520041$, Key Program Special Fund in XJTLU ($KSF-A-01$).

Compliance with Ethical Standards

Conflict of Interests The authors declare that they have no conflict of interest.

Ethical Approval This article does not contain any studies with human participants performed by any of the authors.

References

1. Adams RP, Wallach HM, Ghahramani Z. Learning the structure of deep sparse graphical models. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics; 2010. p. 1–8.
2. Arnold L, Ollivier Y. Layer-wise learning of deep generative models. CoRR arXiv:1212.1524; 2012.
3. Baek J, McLachlan GJ. Mixtures of common t-factor analyzers for clustering high-dimensional microarray data. Bioinformatics. 2011;27(9):1269–1276.
4. Baek J, McLachlan GJ, Flack LK. Mixtures of factor analyzers with common factor loadings: applications to the clustering and visualization of high-dimensional data. IEEE Trans Pattern Anal Mach Intell. 2010;32(7):1298–1309.
5. Bengio Y. Learning deep architectures for AI. Found Trends Mach Learn. 2009;2(1):1–127.
6. Chen B, Polatkan G, Sapiro G, Dunson DB, Carin L. The hierarchical beta process for convolutional factor analysis and deep learning. In: Proceedings of the 28th International conference on machine learning; 2011. p. 361–368.
7. Everett B. An introduction to latent variable models. Springer Science & Business Media; 2013.
8. Ghahramani Z. Probabilistic machine learning and artificial intelligence. Nature. 2015;521(7553):452.
9. Ghahramani Z, Hinton G. The em algorithm for mixtures of factor analyzers. In: Technical Report CRG-TR-96-1. University of Toronto; 1996. p. 11–18. http://www.gatsby.ucl.ac.uk/.zoubin/papers.html.
10. Hinton GE, Osindero S, Teh YW. A fast learning algorithm for deep belief nets. Neural Comput. 2006;18(7):1527–1554.
11. Hinton GE, Salakhutdinov RR. Reducing the dimensionality of data with neural networks. Science. 2006;313(5786):504–507.
12. Jiang Z, Zheng Y, Tan H, Tang B, Zhou H. Variational deep embedding: an unsupervised and generative approach to clustering. In: Proceedings of the twenty-sixth international joint conference on artificial intelligence; 2017. p. 1965–1972.
13. Johnson B. High resolution urban land cover classification using a competitive multi-scale object-based approach. Remote Sens Lett. 2013;4(2):131–140.
14. Johnson B, Xie Z. Classifying a high resolution image of an urban area using super-object information. ISPRS J Photogramm Remote Sens. 2013;83:40–49.
15. Kung SY, Mak MW, Lin SH. Biometric authentication: a machine learning approach, chap. Expectation-maximization theory. Upper Saddle River: Prentice Hall Professional Technical Reference; 2005.
16. Lecun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proc IEEE. 1998;86(11):2278–2324.
17. Likas A, Vlassis N, Verbeek JJ. The global k-means clustering algorithm. Pattern Recogn. 2003;36(2):451–461.
18. McLachlan G, Krishnan T. The EM algorithm and extensions. Wiley; 2007. vol. 382.

19. McLachlan GJ, Peel D. Mixtures of factor analyzers. In: International Conference on machine learning (ICML); 2000. p. 599–606.
20. Nene SA, Nayar SK, Murase H. Columbia object image library (coil-20). Tech. rep. Technical Report CUCS-005-96. 1996.
21. Patel AB, Nguyen T, Baraniuk RG. A probabilistic theory of deep learning. arXiv:1504.00641. 2015.
22. Rippel O, Adams RP. High-dimensional probability estimation with deep density models. CoRR arXiv:1302.5125. 2013.
23. Salakhutdinov R, Mnih A, Hinton GE. Restricted boltzmann machines for collaborative filtering. In: Machine learning, proceedings of the twenty-fourth international conference (ICML); 2007. p. 791–798.
24. Tang Y, Salakhutdinov R, Hinton GE. Deep mixtures of factor analysers. In: Proceedings of the 29th international conference on machine learning. ICML; 2012.
25. Tortora C, McNicholas PD, Browne RP. A mixture of generalized hyperbolic factor analyzers. Adv Data Anal Classif. 2016;10(4):423–440.
26. Wang W. Mixtures of common factor analyzers for high-dimensional data with missing information. J Multivar Anal. 2013;117:120–133.
27. Wei H, Dong Z. V4 neural network model for shape-based feature extraction and object discrimination. Cogn Comput. 2015;7(6):753–762.
28. Wen G, Hou Z, Li H, Li D, Jiang L, Xun E. Ensemble of deep neural networks with probability-based fusion for facial expression recognition. Cogn Comput; 201. https://doi.org/10.1007/s12559-017-9472-6.
29. Yang X, Huang K, Goulermas JY, Zhang R. Joint learning of unsupervised dimensionality reduction and gaussian mixture model. Neural Process Lett. 2017;45(3):791–806.
30. Yang X, Huang K, Zhang R. Deep mixtures of factor analyzers with common loadings: aa novel deep generative approach to clustering. In: Neural Information processing - 24rd international conference, ICONIP; 2017.
31. Zeng N, Wang Z, Zhang H, Liu W, Alsaadi FE. Deep belief networks for quantitative analysis of a gold immunochromatographic strip. Cogn Comput. 2016;8(4):684–692.
32. Zhang J, Ding S, Zhang N, Xue Y. Weight uncertainty in Boltzmann machine. Cogn Comput. 2016;8(6):1064–1073.
33. Zheng Y, Cai Y, Zhong G, Chherawala Y, Shi Y, Dong J. Stretching deep architectures for text recognition. In: Document Analysis and recognition (ICDAR)–13th international conference. IEEE; 2015. p. 236–240.
34. Zhong G, Yan S, Huang K, Cai Y, Dong J. Reducing and stretching deep convolutional activation features for accurate image classification. Cogn Comput. 2018;10(1):179–186.