Neurocomputing 433 (2021) 199–211

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Unsupervised deep clustering via adaptive GMM modeling and optimization

Jinghua Wang, Jianmin Jiang *

Research Institute for Future Media Computing, College of Computer Science & Software Engineering, and Guangdong Laboratory of Artificial Intelligence & Digital Economy (SZ), Shenzhen University, China

ARTICLE INFO

Article history:
Received 31 December 2019
Revised 10 June 2020
Accepted 18 December 2020
Available online 5 January 2021
Communicated by Zenglin Xu

Keywords:
Clustering
Deep learning
Representation learning
Unsupervised learning
Gaussian Mixture Model

ABSTRACT

Supervised deep learning techniques have achieved success in many computer vision tasks. However, most deep learning methods are data hungry and rely on a large number of labeled data in the training process. This work introduces an unsupervised deep clustering framework and studies the discovery of knowledge from a set of unlabeled data samples. Specifically, we propose a new network structure for both representation learning and GMM (Gaussian Mixture Model)-based representation modeling. In the training process of our proposed network, we not only adjust the Gaussian components to better model the distribution of representations, but also adjust the data representations towards their associating Gaussian centers to provide more adaptive support for the GMM. In this way, we take the data representations as the supervisory signal for the update of the GMM parameters and the GMM as the supervisory signal for the update of the representations, yet keeping the entire deep clustering as unsupervised. Consequently, we train the network based on an objective function with two learning targets. With the first target, we learn a GMM to model the representations properly and make each Gaussian component to be compact as much as possible. With the second target, we improve the inter-cluster distance by adapting the cluster centers to be further away from their neighbors. Thus, the training procedure simultaneously improves the intra-cluster compactness and inter-cluster separability for all the evolved clusters. Experimental results on eight datasets show that the proposed method can improve the clustering performance in comparison with the existing state of the art techniques.

1. Introduction

Recent development on machine learning has illustrated that we can now process explosive increase of data samples more effectively with deep learning techniques, and significant advancements have been made across a range of subject areas [1–4]. In the past years, deep learning has been successfully applied in many tasks, such as object recognition [5,6], metric learning [7,8], and object detection [9,10]. However, these models involve many parameters and require a huge number of labeled training samples. In an era of data explosion, we can easily collect varieties of data, including images, texts, and videos. Yet, it is very time-consuming to manually label these data. Thus, it becomes crucial to develop unsupervised techniques for knowledge discovery from the unlabeled data.

As a popular unsupervised learning task, *clustering* categorizes data samples into a number of groups. Many methods determine the clustering results based on the similarities between sample pairs in the original sample space [11]. Thus, the data sample distribution can significantly influence clustering performances [12]. Taking k-means as an example, it works well only if the data samples are distributed around a number of center points.

A procedure of feature extraction or representation learning can improve the robustness of clustering methods against data sample distribution [13–16]. The generic representations can be applied in many different tasks and achieve better performances [17–19]. However, these representations are not necessarily optimal for the task of clustering. To further improve the clustering performance, researchers [20–27] propose to integrate the deep representation learning with clustering into a single framework. These approaches normally first assign a soft label for each data sample, then iteratively refine the soft label based on well-defined objectives until convergence [22].

In this paper, we propose a novel network structure that can jointly optimize clustering and deep representation learning. In

* Corresponding author.
E-mail addresses: wang.jh@szu.edu.cn (J. Wang), jianmin.jiang@szu.edu.cn (J. Jiang).

https://doi.org/10.1016/j.neucom.2020.12.082
0925-2312/

its training procedure, our method simultaneously improves the intra-cluster compactness and inter-cluster separability for all the evolved clusters. This provides a new way to improve the clustering performance. Specifically, we use a GMM to capture the distribution of the whole data representations (from many different clusters) and expect each Gaussian component to represent a cluster. In this way, our method has a broader application scope than the existing approaches [28,29], which formulate the distribution of data representations by a single Gaussian component.

To capture the distribution of the data representations using a GMM, the common approach is to maximize the GMM-likelihood and thus discover a feasible GMM. For the tasks of distribution modeling with fixed data representations, a larger GMM-likelihood means a better GMM to fit the representations. As seen in Fig. 1(a), we can achieve the best GMM with maximized GMM-likelihood to model the given data samples. An important but rarely mentioned point in GMM modeling is that we can also increase the GMM-likelihood by adjusting the data representations towards their associating Gaussian centers, as shown in Fig. 1(b). In other words, the GMM-likelihood can facilitate not only the GMM modeling but also the representation learning.

In our work, both the GMM parameters and the data representations are iteratively updated. In other words, the proposed training procedure not only adjusts the Gaussian components to better model the representation distribution, but also adjusts the data representations towards their associating Gaussian centers to follow the GMM. Intuitively, we take the data representations as the supervisory signal for the update of GMM parameters and the GMM as the supervisory signal for the update of representations. In this way, we not only position the Gaussian components well in the representation space, but also enhance the compactness of each Gaussian component.

In a clustering task, we also expect the clusters to be separable from each other as much as possible. To achieve this, we explicitly maximize the distance between the Gaussian centers, which is achieved by updating the Gaussian centers to make them far away from each other, as shown in Fig. 1(c). By doing this, we can enhance the separability of the clusters, and implicitly enlarge the inter-cluster distances between data sample representations.

We implement our idea with a new network structure by integrating an auto-encoder with a representation modeling network (RMN). The auto-encoder uses an encoder to learn representations from data samples and a decoder to reconstruct the samples from their representations. The newly introduced RMN consists of two layers (*i.e.*, the Λ layer and the G layer) and has a one-to-one correspondence with the GMM. While a node in Λ layer represents a Gaussian component, a node in Glayer represents the Gaussian likelihood. Our method jointly optimizes the RMN with both the encoder and the decoder in order to improve the clustering performance.

In summary, our contributions can be highlighted as:

- In contrast to the traditional GMM-based methods that update the GMM parameters to model a set of fixed data representations, we propose a joint optimization method that iteratively updates both the data representations and the GMM parameters. To the best of our knowledge, such joint optimization of both data representation learning and GMM modeling has not been studied in the area of unsupervised clustering.
- To achieve a joint optimization, we propose a new deep clustering network structure, consisting of an encoder, a decoder, and a representation modeling network (RMN). We also introduce an objective function and a learning method for the proposed network. The newly proposed RMN elegantly represents the GMM parametrically by a Λ layer and a G layer, both of which are initializable with a GMM. The RMN is jointly trainable with the representation learning sub-networks and performs well on both balanced and imbalanced datasets.
- With our newly proposed objective function, our method can improve both the intra-cluster compactness and the inter-cluster separability, leading to improved unsupervised clustering.

The rest of the paper is organized as follows. Section 2 presents the research works. Section 3 details the proposed method. Section 4 presents comprehensive experiments and result analysis. Section 5 concludes the research work described in the paper.

2. Related work

2.1. Clustering

In general, clustering algorithms developed so far can be categorized into two groups, *i.e.*, partitional algorithms and hierarchical algorithms. Hierarchical clustering algorithms [30] build a tree structure of the clusters, either dividing larger clusters into smaller ones or merging smaller clusters into larger ones. Partitional algorithms, such as k-means [31] and Gaussian Mixture Model (GMM) [32], find a division of the data samples into non-overlapping clusters.

Most algorithms cannot achieve satisfying performances if directly conducted on the data space [12]. Thus, researchers propose to improve the clustering performance by transforming the data samples into a feature space [13,14]. To extract discriminant features for clustering tasks, researchers have developed a range of different objective functions, such as sparse self-

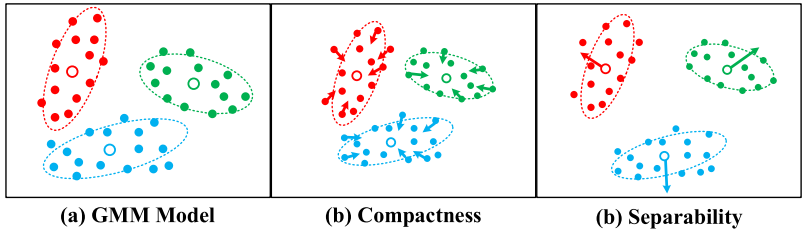


Fig. 1. An intuitive example of our proposed method. While the solid points in different colors denote the data samples from different clusters, the circles denote the centers of different Gaussian components. (a) models data distribution via traditional GMM method with three Gaussian components. (b) improves intra-cluster compactness by reducing the distances between representations and their associating Gaussian centers. The update direction is marked by the arrows. This procedure can also increase the GMM-likelihood. (c) improves inter-cluster separability by adapting the Gaussian centers to be further away from each other. Our method not only updates the data representations to improve the compactness of the clusters (as shown in (b)), but also updates the Gaussian centers to enhance the separability (as shown in (c)).

expressiveness [33], orthogonal matching pursuit [34], and non-negative matrix factorization (NMF) [35–37]. Huang et al. [35] propose an adaptive graph regularized NMF to jointly optimize similarity learning and matrix decomposition. Pan and Gillis [36] generalize the separability assumption of NMF and provide a convex optimization algorithm. Huang et al. [37] introduce a robust graph regularized NMF model to deal with the sparse corruption in the data matrices. Based on the low-level or hand-crafted features, the clustering methods can achieve good performances on a limited number of tasks [38–40].

To overcome the limitation of low-level features, researchers adopt deep learning techniques to extract deep representations and improve the clustering performance [41]. Typical examples include autoencoders [15] and deep belief networks (DBN) [16], which learn deep representations in the task of reconstruction. Other methods learn deep representations based on some certain supervisory signals. FaceNet [17] accomplishes representation learning by analyzing the distance between face images. Wavelet scattering network [18] learns deep representations that are robust against deformation. In addition, researchers also take the context [42] and image sequence order [43] as supervisory signals for deep representation learning.

The generic deep representations can be applied in many different tasks and achieve better performance than the low-level features [15]. However, these representations are not necessarily optimal for the task of clustering. To further improve the clustering performance, researchers [44] propose to integrate the deep representation learning with clustering in a single framework. Different from the work [42,43] where the supervisory signals are fixed, these methods can adapt the cluster assignments and jointly optimize them together with the representation learning. Ren et al. [45] propose a density-based technique where the deep representations are projected to a two-dimensional space via t-SNE. Based on the idea of agglomerative clustering, Yang et al. [20] propose a recurrent framework consisting of two steps, *i.e.,* clustering with fixed representations and supervised representation learning. Ghasedi et al. [44] extend generative adversarial network by a clusterer to learn representations with large depth. Xu et al. [46] propose invariant information clustering (IIC) that can produce semantic clusters. In order to take full advantage of various kinds of correlations, Wu et al. [47] introduce a network for deep comprehensive correlation mining. Dizaji et al. [23] introduce a clustering objective function based on KL-divergence and use an iterative strategy for optimization. To achieve joint optimization, Yang et al. [21] introduce an objective function consisting of three parts, *i.e.,* dimension reduction, data reconstruction, and cluster structure regularization. It can successfully discover clustering-friendly latent representations. However, this work only considers the intra-cluster compactness and ignores the inter-cluster separability. The work that mostly related to our proposed method is DEC [22], which first introduces a soft cluster assignment based on an auxiliary distribution, and then iteratively refines the soft assignment. Guo et al. [48] adopt data augmentation to improve the performance of DEC. Ren et al. [49] extend DEC into a semi-supervised method in order to guide the training procedure with prior knowledge. DEC-based method can achieve good performance, but only in the cases where the initial soft cluster labels are mostly correct. For imbalanced data sets, however, the method will fail, details of which are discussed in subSection 3.3.

2.2. Representation modeling via GMM

Many methods model the representations as a pre-defined distribution, and achieved good performances in a variety of tasks, such as scene recognition [50] and object recognition [51]. Because of its soundness in theory, GMM is one of the most popular methods in representation modeling among various pre-defined distributions. It is theoretically proved that, with certain constraints on its parameters, GMM degenerates to some other models, including k-means [11] and Gaussian-softmax restricted Boltzmann machines [52]. Thus, GMM can be considered as a generalization of these methods.

Modeling the representations by GMM has been extensively studied in the area of automatic speech recognition [53–55]. Theoretically, a single Gaussian model with a globally pooled covariance matrix is equivalent to the soft-max layer in CNN (convolutional neural network) [56]. Based on GMM modeling, Paulik [57] proposes a method for bottleneck representation learning. However, this work cannot guarantee an optimal GMM as it does not update the GMM parameters in its learning process. Tuske et al. [58] integrate GMM into deep neural networks based on the equivalence of log-linear mixture model and GMM [59].

Recently, GMM-based methods are also widely applied in the vision community. Jiang et al. [60] combine GMM and the variational autoencoder for unsupervised representation learning. This work assumes a latent space for the data samples to follow a GMM, and trains a neural network to produce observable data. To extend regular variational autoencoder which uses a single Gaussian component as a prior, Dilokthanakul et al. [61] propose to use a GMM as the prior and thus generate samples belonging to different classes. Graves [62] theoretically studies the algorithm for backpropagation through GMMs. Van den Oord and Benjamin [63] extend the GMM to multiple layers and propose deep GMM to model complex visual features.

This work models the data representations of the whole dataset as a GMM, and expects the data representations from the same cluster to share a Gaussian component. Different from the methods that use GMM to model fixed data representations, our method not only adapts the GMM to better model the representations, but also takes the GMM as a supervisory signal to improve the representations. In comparison with our previous work [64], this paper provide many more theoretical analysis, implementation details and experimental results.

3. The proposed deep clustering

In this section, we first show how we model the data representations using GMM and define the objective function. We then introduce the neural network structure that can learn proper deep representations for clustering. After that, we present our discussion on the advantages of the proposed method.

3.1. Representation learning and GMM-based modeling

As the clustering results are heavily influenced by the data sample representations, we propose to integrate representation learning and clustering into a single framework and jointly optimize them. This subsection first derives a new objective function for representation modeling, then presents its gradients regarding the GMM parameters and the representations.

Different from the supervised techniques that derive the representations based on their correlation with the class labels, our method focus on the distribution of representations. Specifically, we propose to capture the distribution of the representations from unlabeled data samples by a GMM and expect that the representations from the same cluster share a Gaussian component. GMM is theoretically elegant and equivalent to many other models with particular constraints on its parameters [11,52]. Given a set of samples, Stuhlsatz et al. [28] show that a neural network can extract the representations that follow Gaussian distribution. Thus, it is theoretically sound to model deep representations with a GMM.

This is also validated by the fact that Gaussian distributed representations are widely used to solve visual recognition problems [29,50].

Let H denote the dimension of the input x and $f_\theta : R^H \rightarrow R^D$ denote the mapping from the data sample to its representation, where f_θ is a convolutional neural network with θ as the parameter and D is the dimension of the representation. Thus, $f_\theta(x)$ is the representation of the data sample x. We model the distribution of $f_\theta(x)$ as a GMM:

$$p(f_\theta(x)|\lambda) = \sum_{k=1}^{m} \omega_k g(f_\theta(x)|\mu_k, \Sigma_k) \tag{1}$$

where $m$is the number of Gaussian components and $g(z|\mu_k, \Sigma_k)(k = 1, \cdots, m)$ represent the Gaussian densities. The parameters $\omega_k(k = 1, \cdots, m)$are the positive mixture weights that satisfy $\Sigma_{k=1}^{m}\omega_k = 1$. Each Gaussian component density is a Gaussian function, *i.e.,*

$$g(f_\theta(x)|\mu_k, \Sigma_k) = \frac{1}{\sqrt{(2\pi)^D|\Sigma_k|}} exp\left\{-\frac{1}{2}(f_\theta(x) - \mu_k)^T\Sigma_k^{-1}(f_\theta(x) - \mu_k)\right\} \tag{2}$$

where μ_k is the mean vector and Σ_k is the covariance matrix.

For simplicity, let λ denote a combinational group of the mean, covariance, and the mixture weight of the Gaussian components, *i.e.,* $\lambda = \{\omega_k, \mu_k, \Sigma_k, k = 1, \cdots, m\}$. One key problem of GMM-based method is to estimate the parameter λ that best matches the distribution of the training data. In order to estimate the parameters, we normally maximize the GMM-likelihood formulated as follows:

$$p(f_\theta(X)|\lambda) = \prod_{n=1}^{N} p(f_\theta(x_n)|\lambda) \tag{3}$$

where $X = \{x_1, x_2, \ldots, x_N\}$ represents the whole data sample set and $N$is the number of data samples. Let $f_\theta(X) = \{f_\theta(x_1), f_\theta(x_2), \ldots, f_\theta(x_N)\}$ be the representations of all the data samples. In the cases where the representations are fixed, the expectation–maximization (EM) algorithm estimates the parameters by alternatively conducting two different steps (including *an expectation step and a maximization step*) until a convergence is reached. Based on the fixed parameter λ, the *expectation step* assigns a posterior probability for every sample belonging to each Gaussian component. With the posterior probability (calculated in the *expectation step*) of each sample, the *maximization step* updates the parameter of each Gaussian component in order to achieve the maximized GMM-likelihood (Eq. (3)).

To introduce the principle of joint optimization of both deep representation learning and GMM-based representation modeling, we propose an objective function as follows:

$$\begin{aligned} O &= log(P(f_\theta(X)|\lambda)) + \eta S(\mu) \\ &= log\left(\prod_{n=1}^{N} p(f_\theta(x_n)|\lambda)\right) + \eta\sum_{k=1}^{m}\sum_{j\in n(k)} dis(\mu_k, \mu_j) \end{aligned} \tag{4}$$

The parameter $\eta > 0$ is non-negative, which is used to balance the two terms. For the kth Gaussian component, we use $n(k)$ to denote its neighboring Guassian components in the GMM and identify the neighbors based on the normalized distance $dis(\mu_k, \mu_j)$ between Gaussian centers. While the first term in Eq. (4) evaluates the log GMM-likelihood of the representations $f_\theta(X)$, the second term calculates the distances among the Gaussian centers. Both of the two terms produce scalar values and thus can be jointly optimized. Essentially, we can obtain the best possible GMM by maximizing the first term, and enhance the separability of Gaussian components by maximizing the second term.

The first term in Eq. (4) has two sets of parameters, *i.e.,* the neural network parameter θ and the GMM parameter λ. While λ determines the distribution of representations, θ determines the representations themselves. To achieve joint optimization, we iteratively update λ to better model the distribution of representations, and update θ in order to adjust the representations towards their associating centers, leading to enhanced compactness of each Gaussian component. In summary, maximizing the first term in Eq. (4) guarantees that: 1) the learned GMM can best capture the distribution of representations; and 2) each Gaussian component corresponds to a compact cluster.

In addition to the intra-cluster compactness guaranteed by the first term, inter-cluster separability is another important criterion for clustering [65]. The second term in Eq. (4) involving the GMM parameter λ aims to enhance the inter-cluster separability. This is achieved by enlarging the distance between Gaussian centers, and thus implicitly increase the distance between representations associating different Gaussian components in the following iterations. This term also achieves another advantage that trivial solutions can be avoided, in which all the samples share the same representation.

To optimize our proposed framework for CNN-based representation learning and GMM modeling, we update the parameters θ and λ iteratively based on the evaluations of data samples. With a data sample x as the input, specifically, we introduce and maximize the following objective function

$$O(x|\lambda, \theta) = log(p(f_\theta(x)|\lambda)) + \eta\sum_{k=1}^{m}\sum_{j\in n(k)} \|\mu_k - \mu_j\|^2 \tag{5}$$

With D denotes the dimensionality of the representations, we restrict the covariance matrix to be diagonal, *i.e.,* $\Sigma_k = diag(\sigma_{k1}^2, \sigma_{k2}^2, \cdots, \sigma_{kD}^2)$, which are justified as follows. Firstly, a higher-order diagonal covariance matrix can theoretically approximate any full covariance matrices in GMM [66]. Secondly, a diagonal covariance matrix can speed up the learning procedure due to its fewer parameters and efficient inversion matrix computing method. As indicated in Eq. (8), we need the inversion of covariance matrix to update the network parameters, and it is computationally very expensive to compute the inversion of a full covariance.

Let y be the representation of x, *i.e.,* $y = f_\theta(x)$. We can work out the following derivation of the objective function regarding y:

$$\begin{aligned} \frac{\partial O(x|\lambda,\theta)}{\partial y} &= \frac{\partial log(p(y|\lambda))}{\partial y} = \frac{1}{p(y|\lambda)}\frac{\partial(p(y|\lambda))}{\partial y} \\ &= \frac{1}{p(y|\lambda)}\sum_{k=1}^{m}\omega_k\frac{\partial g(y|\mu_k,\Sigma_k)}{\partial y} = \sum_{k=1}^{m}\frac{\omega_k}{p(y|\lambda)}\frac{\partial g(y|\mu_k,\Sigma_k)}{\partial y} \end{aligned} \tag{6}$$

As $\frac{\partial log f(x)}{\partial x} = \frac{1}{f(x)}\frac{\partial f(x)}{\partial x}$, we have

$$\frac{\partial log g(y|\mu_k, \Sigma_k)}{\partial y} = \frac{\Sigma_k^{-1} + (\Sigma_k^{-1})^T}{2}(\mu_k - y) \tag{7}$$

With a diagonal covariance matrix, we have

$$\frac{\partial O(x|\lambda, \theta)}{\partial y} = \sum_{k=1}^{m} p(c_k|y)\Sigma_k^{-1}(\mu_k - y) \tag{8}$$

where,

$$p(c_k|y) = \frac{\omega_k g(y|\mu_k, \Sigma_k)}{p(y|\lambda)} \tag{9}$$

denotes the probability that sample x_i belongs to the kth Gaussian component.

Similarly, we can work out the following derivatives as given in Eqs. (10)–(12) for the objective function regarding the parameters in Eq. (5)

$$\frac{\partial O(x|\lambda,\theta)}{\partial \mu_{kd}} = p(c_k|y)\frac{y_d - \mu_{kd}}{\sigma_{kd}^2} + 2\eta \sum_{j\in n(k)} (\mu_{kd} - \mu_{jd}) \tag{10}$$

$$\frac{\partial O(x|\lambda,\theta)}{\partial \sigma_{kd}} = p(c_k|y)\left[\frac{(y_d - \mu_{kd})^2}{\sigma_{kd}^2} - 1\right] \tag{11}$$

$$\frac{\partial O(x|\lambda,\theta)}{\partial \omega_k} = p(c_k|y) - \omega_k \tag{12}$$

Here, the index $1 \leqslant k \leqslant m$ corresponds to the Gaussian components, and $1 \leqslant d \leqslant D$ corresponds to the dimensionality of the representations (or the parameters). Specifically, μ_{kd} and y_d respectively denote the dth dimension of the kth mean vector and the representation. The training procedure updates the parameter θ based on Eq. (8) by back-propagation and simultaneously updates the representation y. Our training procedure also updates the GMM parameters based on Eqs. (10)–(12).

3.2. Deep clustering network

We propose a network structure (as shown in Fig. 2(a)) consisting of three components, *i.e.,* the encoder, the decoder, and the representation modeling network (RMN). While the encoder extracts deep representations from the data samples, the decoder reconstructs the data samples based on the extracted representations, and the RMN adaptively models the distribution of data representations by a GMM. As shown in Table 1, we train the proposed network in three stages with different objectives. Each stage only adapts the parameters of one or two components in the whole network.

The first stage trains the encoder and the decoder together to learn representations in a reconstruction task. It is recognized that the encoder outputs semantically meaningful and well separated representations on real-world images [15]. At this stage, the input data sample is randomly corrupted first, and then a denoising autoencoder is used to reconstruct the clean sample. The objective is to minimize the squared distance between the input data sample $x$and the reconstructed result x_r, *i.e.,* $||x - x_r||^2$. We also adopt the data augmentation technique proposed in [48] to learn more representative features. Our first training stage has three steps. The first step trains a number of RBMs (restricted boltzmann machines) in a greedy manner. The second step unfolds the RBMs and concatenates them together to construct the encoder and the decoder. The third step fine-tunes the whole autoencoder based on the reconstruction task.

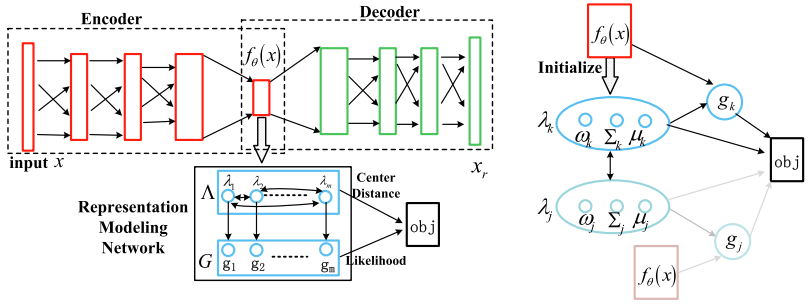
The second stage primarily initializes the parameters of RMN, a new structure to represent a GMM. As seen in Fig. 2, the RMN consists of two layers, one is represented by $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_m\}$ and the other by $G = \{g_1, g_2, \ldots, g_m\}$. We initialize the Λ layer with two steps. The first step extracts representations for all the data samples using the encoder and obtains a representation set. The second step learns a GMM to capture the distribution of the representations, and assigns values for the λ nodes. Specifically, the node $\lambda_k = \{\mu_k, \Sigma_k, \omega_k\}$ represents the kth Gaussian component with three values, *i.e.,* the Gaussian center, the covariance, and the weight. With the λ_k node and a representation, we can obtain the value of node $g_k (1 \leqslant k \leqslant m)$ by Eq. (2), *i.e.,* the Gaussian likelihood. Both the two layers in RMN take the data representations produced by the encoder (denoted as $f_\theta(x)$ in Fig. 2(b)) as the input.

The third stage jointly optimizes the encoder and RMN based on the proposed objective function. It simultaneously fine-tunes the encoder to produce more clustering-friendly representations, and adapts the RMN to better model the representation set. The parameters of both the encoder and RMN are updated based on Eqs. (8) and (10)–(12).

3.3. Discussion

Although unsupervised deep clustering has been widely researched over the past years, the work that most relates to our proposed is DEC [22], which jointly optimizes deep representation learning and k-means modeling. Compared with DEC, the proposed method has the following two advantages.

Firstly, by modeling deep representations with a GMM, our method is sound in theory and applicable to more clustering tasks than DEC [22]. It is recognized that a neural network can extract Gaussian distributed deep representations from an arbitrary data sample set [28]. In theory, we can learn GMM distributed deep representations from a set of samples belonging to a number of clusters, with each Gaussian component corresponding to a cluster. In contrast, k-means adopted by DEC is only a specific case of GMM and thus has limitations in terms of applicable scope. Theoretically, GMM degenerates to k-means, if the covariance matrices are equal to the scalar matrix for all Gaussian components, *i.e.,* $\Sigma_k = \epsilon I$ with



(a) Network Structure **(b) λ node and g node**

Fig. 2. a) Our network structure and b) the relationship between Λ layer and G layer. The network consists of three components, *i.e.,* the encoder, the decoder, and the representation modeling network. The encoder extracts deep representations from the samples. The decoder reconstructs the samples based on the extracted representations. The representation modeling network models the data representations by a GMM. Each node λ represents the weight, the mean and covariance matrix of a Gaussian component. The Λ layer is initialized based on the deep representations $f_\theta(x)$produced by the encoder. Note that, the Gaussian components are mutually influenced by each other. Each node $g_k (1 \leqslant k \leqslant m)$ denotes the value of the kth Gaussian component evaluated by Eq. (2). The objective function is calculated based on the likelihood produced by the G layer and the distance between the centers produced by the Λ layer.

$\epsilon \to 0$ [11]. Here, ϵ denotes a fixed positive real value with small absolute value. Note that, when $\epsilon \to 0$, we need to take heuristic strategies to avoid singularities to make GMM work properly, *i.e.,* no data sample is exactly equal to any Gaussian mean [11]. Based on these observations, we know that modeling the deep representations with a GMM is sound in theory and more general than k-means.

Secondly, DEC is incompetent in dealing with imbalanced datasets. We find that DEC intrinsically tends to produce balanced clusters. In its optimization procedure, DEC defines soft assignments q_{ij} and tries to match them with the following target distribution

$$p_{ij} = \frac{q_{ij}^2/f_j}{\sum_{j'} q_{ij'}^2/f_{j'}} \tag{13}$$

where q_{ij} and p_{ij} respectively denote the soft assignment and target probability of assigning the ith sample into the jth cluster. Let $f_j = \sum_i q_{ij}$ denote the expectation of the size of the jth cluster. Here, we assume q_{ir} to be the dominant assignment of sample i, *i.e.,* $q_{ir} \geqslant q_{ij}$ for any j. In other words, the sample $i$is expected to be clustered into the rth cluster based on the soft assignment. By adapting the soft assignment q_{ij} towards the target assignment p_{ij}, DEC claims to refine the cluster assignment by increasing q_{ir} and decreasing $q_{ij}, \forall j \neq r$. However, this is not true for an imbalanced dataset. Let the expected size of cluster r be α times larger than that of cluster e based on the soft assignments, *i.e.,* $f_r = \alpha f_e$, then we can establish the following equation

Table 1
Three stages to train the proposed network.

	Involved Components	Objective
Stage 1	Encoder & Decoder	Minimize $\|x - x_r\|^2$
Stage 2	RMN	Maximize GMM-Likelihood Eq. (3)
Stage 3	Encoder & RMN	Maximize the proposed Eq. (4)

$$p_{ie} - p_{ir} = \frac{q_{ie}^2/f_e - q_{ir}^2/f_r}{\sum_{j'} q_{ij'}^2/f_{j'}} = \frac{q_{ie}^2 - q_{ir}^2/\alpha}{f_e\left(\sum_{j'} q_{ij'}^2/f_{j'}\right)} \tag{14}$$

If $\sqrt{\alpha} q_{ie} > q_{ir}$, we have $p_{ie} > p_{ir}$. For sample i, after matching soft assignment to target distribution, DEC assigns a higher probability to the eth cluster than the rth cluster. This clearly goes against the principle of increasing the dominant assignment and decreasing the rest. Thus, the adapting procedure can destroy the clusters instead of refining them in this imbalanced case. From Eq. (14), it can also be seen that, the smaller the cluster j is, the more likely the target assignment p_{ij} dominates. This indicates DEC always tries to balance the clusters.

4. Experiments

In this section, we report our experimental results on eight publicly available datasets. To show the effectiveness of the proposed method, we use two standard metrics and carry out comparative evaluations by using a range of existing state of the arts clustering techniques as the benchmarks. We explain the implementation details of the proposed method.

[1] https://cs.nyu.edu/ roweis/data.html

4.1. Dataset

To evaluate the proposed, we conduct experiments on eight datasets, *i.e.,* MNIST [67], USPS [1], Fashion [68] CMU PIE [69], COIL 20 [70], COIL 100 [71], STL-10 [72], and REUTERS [73]. Table 2 lists the data statistics, including the size of the dataset, number of classes, and the data dimensionality.

Both MNIST [67] and USPS are digit image datasets. Each image in these two datasets belongs to one of 10 classes, *i.e.,* from 0 to 9. While MNIST has 70k images and USPS has 11k images. The image size is 28×28 in MNIST and 16×16 in USPS. Fashion [68] consists of 70k images from 10 categories. CMU PIE dataset [69] consists of $41,368$ images of 68 people. The images of each person are captured under 43 different illumination conditions, 13 different poses, and with 4 different expressions. COIL20 [70] and COIL100 [71] are built by Columbia University. The COIL20 consists of 72 gray images for each of the 20 objects. COIL100 consists of 72 color images for each of the 100 objects. The images are captured under different views. For all of the above six datasets (*i.e.,* MNIST, USPS, Fashion, CMU PIE, COIL 20, and COIL 100), we take the raw data as the input.

The STL-10 dataset [72] has $1,3000$ labeled image samples from 10 different objects: *airplane, bird, car, cat, deer, dog, horse, monkey, ship*, and *truck*. Each class has 1300 images. This dataset also has $100,000$ unlabeled images, which do not belong to the above 10 classes. The image size is $96 \times 96 \times 3$. We use both labeled and unlabeled datasets to train our autoencoder, and evaluate the clustering performance only on the labeled samples. Following [74], we calculate both the HOG features and the 8×8 color map of each image, and take the concatenation of them as the input (with the dimensionality of $1,428$).

We conduct experiments on two subsets of Reuters [73], *i.e.,* Reuters-4 and Reuters-20. While Reuters-4 consists of $685,071$ documents from 4root topics, Reuters-20 consists of $365,968$ documents from 20non-root topics. Similar to the experiments reported in [75], we use the tf-idf features (with dimensionality of $2,000$) of those most frequently used words to represent the documents. Different from the first seven datasets which are balanced, this dataset is imbalanced and its clusters are significantly different from each other in terms of the cluster sizes. In Reuters-4, the cluster sizes are 299,612 , 54,695 , 163,135, and 167,629. Fig. 3 visualizes the cluster sizes in Reuters-20, where the largest cluster is 13times larger than the smallest one.

4.2. Benchmarks

We compare the proposed method with a number of representative existing unsupervised clustering methods. Firstly, we take K-means [31] and GMM [11] as the benchmarks, which are two of the most popular methods for unsupervised clustering. Both of them can either take the low-level feature or the deep AEF (autoencoder feature) produced by an autoencoder as the input. The AEF is learned for reconstruction and not jointly optimized with any clustering target. We also take the MTD [76] as the baseline, which models the distribution of deep features by a set of t-distributions.

Secondly, we take four agglomerative methods as benchmarks, *i.e.,* agglomerative clustering with average linkage (AC-Link) [77], Zeta function based agglomerative clustering (AC-Zell) [78], graph degree linkage-based agglomerative clustering (AC-GDL) [79], and agglomerative clustering via path integral (AC-PIC) [30].

Thirdly, we compare our method with five subspace-based clustering methods. They are NJW spectral clustering (SCNJW) [13], self-tuning spectral clustering (SC-ST) [14], large-scale spectral clustering (SC-LS) [38], locality preserving non-negative matrix

Table 2
Statistics of the datasets.

Dataset	MNIST	USPS	Fashion	CMUPIE	COIL20	COIL100	STL10	Reuters4	Reuters20
Smp #	70,000	11,000	70,000	41,368	1,440	7,200	13,000	685,071	365,968
Class #	10	10	10	68	20	100	10	4	20
Dim	28×28	16×16	28×28	32×32	128×128	128×128	1,428	2,000	2,000

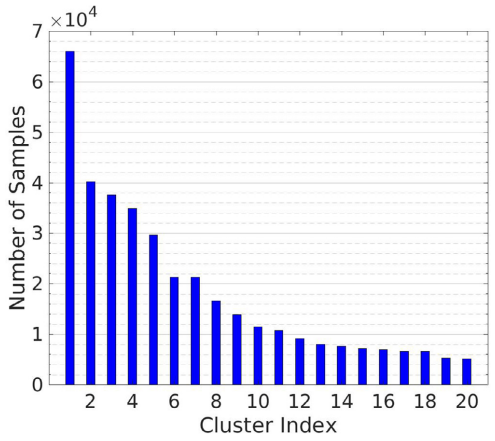


Fig. 3. The number of data samples in each clusters of the Reuters-20. This dataset is imbalanced and the largest cluster is 13 times larger than the smallest one.

factorization (NMF-LP) [39], and NMF with deep model (NMF-D) [40].

Fourthly, we compare our method with seven other benchmarks, which also jointly optimize the representation learning and clustering, *i.e.,* DEC [22], JULE [20], DCN [21], DEPICT [23], ClusterGAN [44], DEC-DA [48], and DGG [27].

Finally, we set the parameter η in Eq. (4) to be 0and produce another benchmark (denoted as DeepGMM in this paper) for ablation study and to assess the effectiveness of the second term of Eq. (4) in improving the separability between Gaussian components. The only difference from the proposed method is that DeepGMM does not explicitly enlarge the distances between the Gaussian centers.

4.3. Implementation details

The first training stage

Our autoencoder contains one encoder and one decoder, each of which has four fully connected layers. At the first training stage, we train the autoencoder in a greedy manner with three steps. In the first step, we train four RBMs in a greedy manner, in which the "hidden" layer of the ith RBM is taken as the "visible" layer of the $(i+1)$th RBM, as shown in Fig. 4(a). We set the number of "hidden" nodes in the first three RBMs to be 500, 500, and 2000, respectively. The number of hidden nodes in the fourth RBM is set to be 10. In the second step, we concatenate the RBMs to build the encoder and the decoder as shown in Fig. 4(b). In the third step, we fine-tune the encoder and the decoder, *i.e.,* all of the 8fully connected layers, to achieve the best possible reconstruction. For an input, we extract features from the last layer of the encoder, *i.e.,* the bottleneck layer with 10nodes. This encoder (for feature extraction) will be further adapted in the third training stage to produce clustering-friendly representations.

In the first training stage, we set the base learning rate to be 0.01 and take the stepping policy to update the learning rate, *i.e.,* divided by 10 for every $10k$ iterations. The batch size is set to be 256 and weight decay is set to be 0. In the first step, we train each RBM with $50k$ iterations and set the dropout rate to be 20%. In the third step, we fine-tune the entire autoencoder with $40k$ iterations.

The second training stage

This stage initializes the RMN based on the GMM that best captures the distribution of data sample representations. Specifically, we first extract the representations of all the data samples using the encoder, and learn a GMM to capture the distribution of these representations via an EM algorithm. We then assign values for the λ nodes, *i.e.,* $\lambda_k = \{\mu_k, \Sigma_k, \omega_k\}$, so that the RMN can model the GMM. In other words, each node λ_k corresponds to the three parameters of one Gaussian component, including the weight, the mean and the covariance. To maintain a fair comparison with the benchmark, the number of the Gaussian components in GMM is made equal to the number of clusters in the dataset.

The third training stage

The third stage adopts the SGD (stochastic gradient descent) to jointly optimize the encoder and the RMN. In our objective function described in Eq. (4), we set the cardinality $|n(k)|$ to be half of the number of clusters. In this way, the center of one cluster is influenced by half of the remaining clusters that are nearby. For the parameter η in Eq. (4), we choose the best one from $\{0.1, 0.01, 0.001, 0.0001\}$. We set the batch size to be 128 and the base learning rate to be 0.01. We take the stepping policy to update the learning rate and train the encoder with RMN for $50k$ iterations.

4.4. Performance

We use two popular metrics to evaluate the clustering algorithms, *i.e.,* clustering accuracy (ACC) [80] and normalized mutual information (NMI) [81]. We conduct clustering for 10 times with different initializations. Tables 3 and 4 respectively list the average clustering accuracy and NMI of the proposed methods in comparison with the benchmarks. The proposed method achieves the best performance (marked by red) in all of the seven image datasets and Reuters-4. This indicates that the proposed method can learn feasible deep representations for various different clustering tasks.

As seen, the deep learning methods perform better than the subspace methods. In all cases, the top 3 performances (marked in bold) on each dataset are achieved by deep learning methods. Based on such observation, we can conclude that the nonlinear deep representations learned by deep models are more feasible for clustering than the linear or low-level features. Thus, it is necessary to develop deep learning methods for unsupervised clustering.

As seen from Tables 3 and 4, though MTD can outperforms the AEF + GMM baseline method on STL-10 and Reuteres-4, its performances are significantly worse than the proposed method on all of the seven datasets. This indicates that the GMM learned by our method can better model the distribution than t-distributions learned by MTD. As known, one of the main advantages of MTD lies its capability in dealing with outliers. However, our representation set has very few outliers due to the robustness of the autoencoder

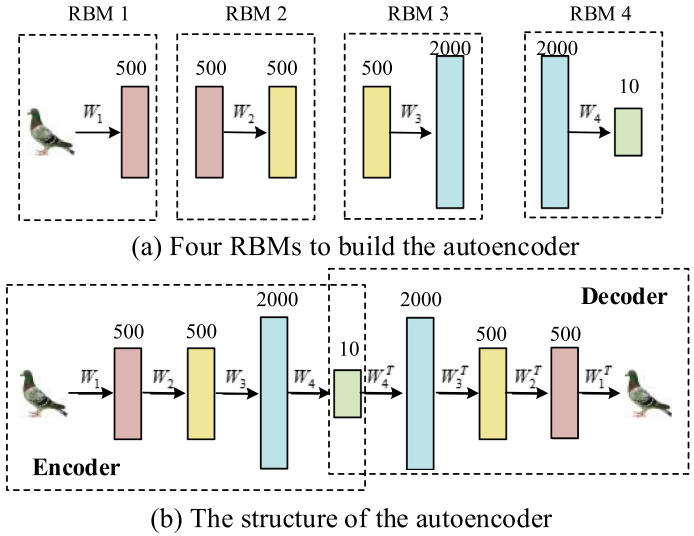


Fig. 4. Illustration of the first training stage, where we train the autoencoder with three steps. The first step trains the four RBMs, the second step builds the autoencoder (*i.e.,* encoder and decoder) based on the trained RBMs, and the third step jointly optimizes the encoder and the decoder to achieve the best possible reconstruction.

Table 3
The accuracy of the proposed method and the baselines on eight datasets (%).

Dataset	MNIST	USPS	Fashion	CMUPIE	COIL20	COIL100	STL-10	Reuters-4
K-Means [31]	53.5±7.1	46.0±5.6	56.3±2.7	54.9±4.9	68.3±7.1	71.4±5.1	28.4±6.1	51.4±4.9
GMM[11]	47.6±5.8	64.2±3.3	48.4±4.5	37.2±3.1	54.3±2.5	67.5±8.2	20.3±4.3	40.5±3.4
AEF+KM	80.0±7.1	64.3±8.4	57.6±4.7	37.2±7.6	54.1±5.0	67.5±5.8	29.7±3.2	42.7±3.4
AEF+GMM	64.1±9.1	71.3±9.3	46.5±2.8	70.3±7.9	69.8±7.0	73.8±5.4	22.2±2.8	43.3±2.4
MTD [76]	60.9±6.3	61.5±6.6	42.0±2.9	58.6±5.8	61.8±6.8	71.5±7.2	30.7±6.1	45.2±4.3
AC-Link [77]	73.2±5.7	66.4±7.5	48.0±4.5	53.7±5.0	49.7±6.7	63.2±5.8	15.3±2.0	59.7±3.9
AC-Zell [78]	8.7±2.7	72.6±7.4	27.9±3.3	84.4±4.5	79.7±5.5	71.4±7.6	16.8±2.6	47.8±5.6
AC-GDL [79]	11.3±2.3	86.7±4.3	33.6±2.1	84.2±6.4	76.5±8.6	70.5±2.3	26.8±1.4	54.6±1.7
AC-PIC [30]	11.5±1.7	85.5±6.5	33.3±4.1	79.7±5.0	70.3±4.9	74.6±3.4	24.1±2.6	65.7±3.6
SCNJW [13]	54.6±4.6	38.6±2.7	32.0±3.6	53.6±4.5	72.8±5.6	72.1±2.7	27.6±4.2	46.9±3.1
SC-ST [14]	31.1±1.5	72.6±3.7	35.6±2.6	58.1±2.6	74.5±4.8	64.1±7.0	12.3±3.0	57.6±2.7
SC-LS [38]	71.4±6.0	65.9±8.3	37.2±1.7	54.9±6.7	76.4±2.5	82.6±5.4	20.4±3.6	60.7±4.6
NMF-LP [39]	43.2±5.9	46.7±7.6	30.9±3.4	63.3±6.6	68.5±7.1	69.5±5.3	11.7±2.3	48.6±2.3
NMF-D [40]	17.5±4.9	38.2±4.6	19.1±4.9	81.0±1.6	64.3±2.4	70.2±7.8	30.6±9.1	38.0±4.8
DEC [22]	84.4±8.0	61.9±7.9	58.7±2.1	80.1±1.2	83.6±9.3	75.5±9.4	35.9±4.5	75.1±2.3
JULE [20]	90.6±1.5	91.4±2.8	59.4±2.1	**89.1±4.6**	80.0±3.5	77.4±9.1	77.7±8.2	75.1±3.4
DCN [21]	83.0±4.2	77.8±6.9	55.2±3.5	82.3±5.6	76.5±6.6	**83.7±5.4**	74.1±5.7	79.2±4.7
DEPICT[23]	**96.5±1.6**	**96.4±1.4**	58.3±3.4	73.4±5.2	63.3±1.5	66.5±4.7	81.8±7.0	75.7±3.6
ClusterGAN[44]	96.4±2.3	**97.0±1.8**	**60.5±2.0**	87.5±1.5	**86.9±2.2**	**84.1±1.9**	**82.3±5.6**	**81.6±3.0**
DGG[27]	**97.6±0.1**	90.4±1.7	**60.6±0.9**	**94.5±2.5**	**93.1±1.6**	81.5±2.0	**90.6±0.2**	**82.3±1.2**
DeepGMM	72.5±6.0	65.4±5.9	45.4±5.9	69.3±5.9	72.3±6.2	52.9±6.9	55.6±2.8	71.8±3.4
Ours	**98.7±0.3**	**98.5±0.2**	**63.5±1.1**	**95.7±1.2**	**94.6±1.5**	**86.3±0.9**	**91.7±1.4**	**84.5±3.2**

in deep feature learning against variations [82]. Thus, MTD cannot show its capability in modeling outliers on the deep representation set. This explains why our method outperforms MTD significantly in our experiments.

In general, our method performs best among all the deep learning methods. In terms of accuracy, our method outperforms the baselines by a margin no less than 1.2% on all of the eight datasets, and the margin can be as large as 2.9% on Fashion and 2.2% on

Table 4
The normalized mutual information (NMI) of the proposed method and the baselines on eight datasets

Dataset	MNIST	USPS	Fashion	CMUPIE	COIL20	COIL100	STL-10	Reuters4
K-Means [31]	0.50±0.03	0.45±0.02	0.34±0.0	0.43±0.04	0.74±0.06	0.78±0.05	0.25±0.05	0.47±0.04
GMM [11]	0.46±0.06	0.63±0.03	0.39±0.0	0.38±0.04	0.51±0.05	0.75±0.04	0.16±0.07	0.43±0.04
AEF+KM	0.73±0.04	0.59±0.03	0.47±0.06	0.58±0.04	0.77±0.05	0.82±0.04	0.26±0.07	0.46±0.03
AEF+GMM	0.59±0.02	0.68±0.04	0.45±0.08	0.72±0.06	0.60±0.05	0.81±0.10	0.20±0.07	0.48±0.05
MTD [76]	0.58±0.08	0.44±0.05	0.36±0.05	0.50±0.10	0.54±0.06	0.66±0.02	0.27±0.04	0.53±0.02
AC-Link [77]	0.69±0.04	0.58±0.03	0.45±0.10	0.55±0.08	0.51±0.04	0.71±0.06	0.14±0.04	0.62±0.05
AC-Zell [78]	0.12±0.01	0.77±0.07	0.32±0.05	0.81±0.05	0.83±0.04	0.79±0.05	0.22±0.04	0.50±0.04
AC-GDL [79]	0.13±0.02	0.82±0.03	0.34±0.03	0.83±0.02	0.80±0.02	0.78±0.02	0.21±0.06	0.49±0.08
AC-PIC [30]	0.09±0.01	0.84±0.03	0.33±0.06	0.80±0.02	0.79±0.04	0.81±0.01	0.18±0.08	0.68±0.02
SCNJW [13]	0.48±0.02	0.41±0.05	0.32±0.05	0.54±0.03	0.76±0.05	0.75±0.04	0.23±0.07	0.55±0.03
SC-ST [14]	0.42±0.04	0.34±0.04	0.27±0.04	0.58±0.06	0.67±0.03	0.71±0.02	0.08±0.01	0.65±0.02
SC-LS [38]	0.71±0.05	0.68±0.03	0.50±0.02	0.79±0.04	0.77±0.02	0.83±0.03	0.16±0.01	0.66±0.03
NMF-LP [39]	0.45±0.02	0.43±0.01	0.31±0.07	0.49±0.04	0.72±0.04	0.78±0.04	0.11±0.03	0.51±0.03
NMF-D [40]	0.15±0.09	0.29±0.08	0.15±0.09	0.84±0.03	0.69±0.02	0.72±0.07	0.24±0.02	0.42±0.02
DEC [22]	0.80±0.04	0.58±0.02	0.60±0.05	0.87±0.03	0.84±0.01	0.79±0.03	**0.82±0.04**	0.73±0.03
JULE [20]	0.87±0.04	0.88±0.01	**0.63±0.02**	0.90±0.02	**0.85±0.02**	**0.83±0.04**	0.74±0.06	0.77±0.04
DCN [21]	0.81±0.03	0.85±0.06	0.59±0.07	0.79±0.05	0.84±0.02	**0.84±0.05**	0.80±0.03	0.81±0.04
DEPICT [23]	**0.92±0.02**	**0.93±0.01**	0.62±0.03	**0.94±0.02**	0.79±0.05	0.74±0.03	0.77±0.03	0.80±0.03
ClusterGAN[44]	**0.92±0.04**	**0.93±0.06**	**0.64±0.03**	0.78±0.03	0.77±0.02	0.80±0.04	**0.83±0.03**	**0.82±0.04**
DGG[27]	0.88±0.06	0.82±0.04	0.61±0.04	**0.92±0.03**	**0.88±0.05**	0.78±0.04	0.81±0.02	**0.82±0.03**
DeepGMM	0.64±0.03	0.51±0.06	0.41±0.05	0.62±0.04	0.74±0.07	0.51±0.04	0.71±0.06	0.75±0.03
Ours	**0.96±0.01**	**0.95±0.01**	**0.69±0.04**	**0.94±0.02**	**0.91±0.02**	**0.91±0.02**	**0.86±0.05**	**0.86±0.04**

Reuters-4. In terms of NMI, our method outperforms the second best by a margin of 0.05 on Fashion, and 0.07 on COIL100. Based on these observations, we can conclude that our proposed method achieves advantages over the existing state of the arts in deep representation learning. This also suggests that our proposed can be potentially applied to other difficult data clustering tasks.

The results in Tables 3 and 4 also illustrate that the proposed method always performs better than DeepGMM, which validates that the introduction of the second term in Eq. (4) can indeed improve the separability of the data representations. Specifically, with the second term, we can update the Gaussian components to make them far away from each other in the third training stage. Thus, the learned data representations are not only intra-cluster compact but also inter-cluster separable. In addition, the proposed method significantly outperforms the benchmark AEF + GMM. Although the autoencoder can extract semantic meaningful representations, these representations are learned for reconstruction purposes. By further jointly optimizing them with the GMM modeling, we can significantly enhance their discriminating ability. In comparison with AEF + GMM, our proposed achieves improvement of more than 20% on six datasets in terms of accuracy. This validates the claim we make in this paper that we can improve the clustering performance by learning deep representations which are catered for the clustering tasks.

We evaluate our method on Reuters-20 to assess its capability in handling imbalanced datasets. We first sort the clusters in a descending order based on the cluster size, and then evaluate clustering methods with the first 4, 8, 12, 16, and 20 clusters. Table 5 lists the performances of the eight baselines and our method. On average, our method performs the best by achieving the highest accuracy in all of the 5cases, and the highest NMI in 4cases. On these imbalanced datasets, our proposed method outperforms the DEC [22] by a large margin in terms of both accuracy ($\geqslant 0.43$) and NMI ($\geqslant 0.48$), indicating that GMM can better model the representations of Reuters-20 than the k-means in DEC [22]. This supports our conclusion in Section 3.3 that DEC [22] cannot perform well with imbalanced clusters.

We compare our method with Guo et al. [48] and Ren et al. [45] on the first three datasets, *i.e.*, MNIST, USPS, and Fashion. While Guo et al. [48] achieves the highest accuracy on USPS, our method performs the best on both USPS and Fashion. On Fashion, for example, our method (63.5%) outperforms Guo et al. [48] (59.4%) by a margin of 4.1% and Ren et al. [45] (61.9%) by a margin of 1.6% in terms of accuracy. In addition, we also take FcDEC-DA reproted in [48] as a baseline for clustering of imbalanced datasets in Table 5. As seen, our method outperforms FcDEC-DA [48] by a margin of 0.456 in terms of average accuracy and 0.510 in terms of average NMI.

An important hyper-parameter in our method is the η to balance the two terms in Eq. (4). Table 6 lists the accuracy of our method on CMU-PIE and STL-10 with different parameter settings. As seen, our method achieves the best performance on CMU-PIE (64 clusters) with $\eta = 0.001$and on STL-10 (10 clusters) with $\eta = 0.01$. In our experiments, it is typical that the best performance is achieved with a relatively small η when the number of clusters is large. Based on Table 6, we can also conclude that a smaller η is better than a larger one, if the best settings is not identified. Take the STL-10 as an example, the accuracy of our method is 88.0% when $\eta = 0.001$, only 3.7% lower than the best setting (*i.e.*, $\eta = 0.01$). However, the accuracy is 16.1% lower than the best setting when $\eta = 0.1$.

4.5. Discussion

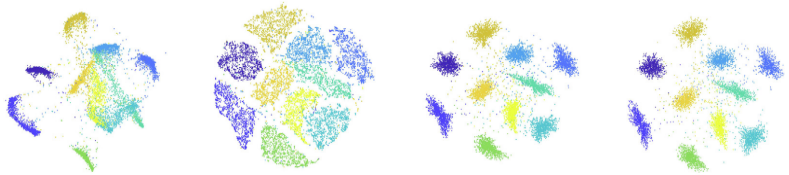
In our approach, we aim to learn compact clusters which are separable. In Fig. 5, we visualize the data representations of a MNIST subset (with 10,000 images) in different epochs by using t-distributed stochastic neighbor embedding (t-SNE) [83]. We can

Table 5
The performance on Reuters-20 with different number of clusters.

Methods	4 Clust.		8 Clust.		12 Clust.		16 Clust.		20 Clust.	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
DEC [22]	0.38	0.11	0.24	0.10	0.18	0.09	0.17	0.09	0.14	0.08
JULE [20]	0.67	0.57	0.59	0.43	0.45	0.41	0.42	0.42	0.35	0.36
DCN [21]	0.80	0.76	0.63	0.63	0.60	0.67	0.51	0.62	0.47	**0.61**
DEPICT [23]	0.78	0.70	0.75	0.67	0.64	0.58	0.61	0.55	0.55	0.49
ClusterGAN[44]	0.73	0.66	0.66	0.61	0.61	0.58	0.56	0.54	0.52	0.46
DGG[27]	0.51	0.50	0.48	0.48	0.39	0.35	0.32	0.29	0.27	0.21
FcDEC-DA [48]	0.39	0.23	0.31	0.17	0.27	0.16	0.18	0.12	0.11	0.08
DeepGMM	0.68	0.60	0.61	0.62	0.67	0.59	0.56	0.56	0.51	0.49
Ours	**0.84**	**0.77**	**0.78**	**0.71**	**0.69**	**0.63**	**0.66**	**0.64**	**0.57**	0.56

Table 6
The performances of our method on CMUPIE and STL10 with different η.

η		0.0001	0.0005	0.001	0.005	0.01	0.05	0.1
CMU-PIE	ACC	0.911	0.950	**0.957**	0.945	0.905	0.861	0.733
	NMI	0.874	0.923	**0.938**	0.897	0.792	0.736	0.675
STL10	ACC	0.831	0.832	0.880	0.896	**0.917**	0.823	0.751
	NMI	0.774	0.770	0.787	0.821	**0.862**	0.714	0.585



(a) Epoch 1 (b) Epoch 4 (c) Epoch 8 (d) Epoch 10

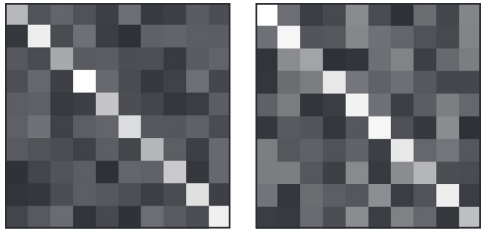
Fig. 5. The representation clusters of MNIST in different epochs. While the initial representations are mixture together, they gradually evolve into separable clusters. In addition, the compactness of the clusters is also gradually improved.

clearly see that the clusters are gradually become more compact and more separable.

To justify the diagonal assumption on covariance matrix, we calculate the covariance matrices of deep features learned by the autoencoder (which are not adapted by the GMM modeling). In Fig. 6, we visualize the covariance matrices of digit "0" and "7" in MNIST. As seen, the diagonal values are much larger than the rest and the matrices are almost diagonal, which justifies our assumption.

To differentiate those difficulty examples from the easier ones, we visualize the data samples which are close to their associating centers and the ones which are far away from the centers (in Fig. 7 for MNIST and in Fig. 8 for the STL-10 dataset). For MNIST, the samples close to the centers are the ones which are similar to the standard written characters. In other words, if the digit is well written, we can easily categorize it into the right cluster. On the contrary, as shown in Fig. 7(b), the images that are far away from their associating centers are not well written (judged by common sense), and most of them are visually distorted. For the digit "9" in the first two rows, as an example, it is difficult even for human beings to recognize it correctly.

In addition to noisy background, the pose of foreground also significantly influences the clustering performances on STL-10 dataset. Our method can correctly cluster those images, in which the target foregrounds are well posed and clearly visible. In principle, it is quite difficult to cluster an image whose foreground is captured in side view or occluded. For example, all the *car* images shown in Fig. 8(b) are mis-clustered due to the heavy occlusions.

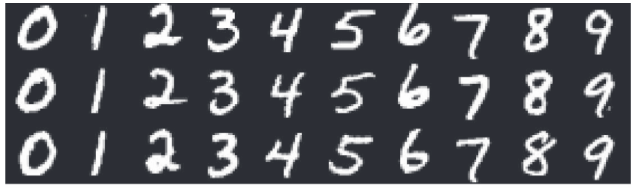


(a) Digit "0" (b) Digit "7"

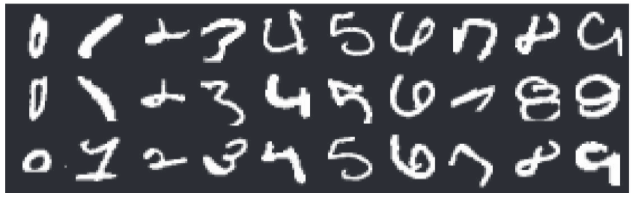
Fig. 6. The covariance matrices of different clusters in MNIST (a) digit "0" and (b) digit "7".

5. Conclusions and future work

This paper formulates an unsupervised deep clustering framework by integrating representation learning and GMM modeling. We achieve the joint optimization by proposing a new network structure as shown in Fig. 2. In order to train the proposed net-



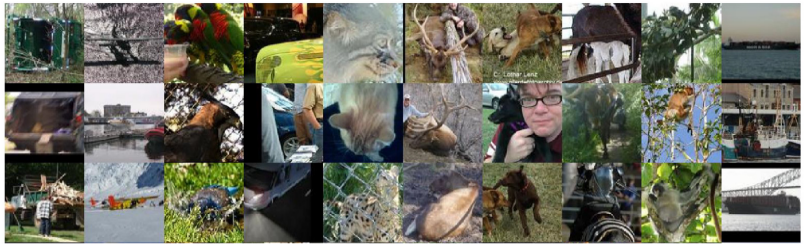
(a) The MNIST images which are close to their associating Gaussian centers



(b) The MNIST images which are far away from their associating Gaussian centers

Fig. 7. Illustration of MNIST images: a) close to their associating centers, and b) far away from their associating Gaussian centers.

(a) The STL-10 images which are close to their associating Gaussian centers



(b) The STL-10 images which are far away from their associating Gaussian centers

Fig. 8. The STL-10 images which are a) close to their associating centers, and b) far away from their associating Gaussian centers. Each of the ten columns represent the images of the same groundtruth cluster. The labels of images are respectively *airplane, bird, car, cat, deer, dog, horse, monkey, ship*, and *truck*.

work, we introduce an objective function with two terms. By optimizing the objective function, our method can improve both intra-cluster compactness and inter-cluster separability. The training procedure not only takes the GMM as the supervisory signal to update the representations, but also takes the representations as signals to update the GMM.

For future research, we will investigate how to model the distribution of the deep features using a mixture of more general components, such as t-distribution or non-elliptical shapes.

Specifically, we will integrate the powerful deep learning technique together with the popular distribution modeling methods in a single framework, and learn representations that follow arbitrary distributions.

CRediT authorship contribution statement

Jinghua Wang: Conceptualization, Methodology, Validation, Writing - original draft. **Jianmin Jiang:** Supervision, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

The authors wish to acknowledge the financial support from: (i) Natural Science Foundation China (NSFC) under the Grant No. 61620106008; (ii) Natural Science Foundation China (NSFC) under the Grant No. 61802266; and (iii) Shenzhen Commission for Scientific Research & Innovations under the Grant No. JCYJ20160226191842793.

References

[1] G. Ciaparrone, F.L. Sánchez, S. Tabik, L. Troiano, R. Tagliaferri, F. Herrera, Deep learning in video multi-object tracking: a survey, Neurocomputing (2019).
[2] D. Wu, S.-J. Zheng, X.-P. Zhang, C.-A. Yuan, F. Cheng, Y. Zhao, Y.-J. Lin, Z.-Q. Zhao, Y.-L. Jiang, D.-S. Huang, Deep learning-based methods for person re-identification: a comprehensive review, Neurocomputing 337 (2019) 354–371.
[3] F. Lateef, Y. Ruichek, Survey on semantic segmentation using deep learning techniques, Neurocomputing 338 (2019) 321–348.
[4] J. Wang, Z. Wang, D. Tao, S. See, G. Wang, Learning common and specific features for RGB-D semantic segmentation with deconvolutional networks, in: ECCV, 2016, pp. 664–679.
[5] X. Zhou, N. Liu, F. Tang, Y. Zhao, K. Qin, L. Zhang, D. Li, A deep manifold learning approach for spatial-spectral classification with limited labeled training samples, Neurocomputing 331 (2019) 138–149.
[6] J.-Y. Kim, S.-B. Cho, Exploiting deep convolutional neural networks for a neural-based learning classifier system, Neurocomputing 354 (2019) 61–70.
[7] L. Ma, H. Li, F. Meng, Q. Wu, K.N. Ngan, Discriminative deep metric learning for asymmetric discrete hashing, Neurocomputing (2019).
[8] D. López-Sánchez, A.G. Arrieta, J.M. Corchado, Visual content-based web page categorization with deep transfer learning and metric learning, Neurocomputing 338 (2019) 418–431.
[9] L.M. Soria, F.J. Ortega, J.A. Álvarez García, F. Velasco, D. Fernández-Cerero, How efficient deep-learning object detectors are?, Neurocomputing (2019)
[10] X. Zeng, L. Wen, B. Liu, X. Qi, Deep learning for ultrasound image caption generation based on object detection, Neurocomputing (2019).
[11] C.M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Springer-Verlag, New York, 2006.
[12] C.C. Aggarwal, C.K. Reddy, Data Clustering: Algorithms and Applications, 1st ed., Chapman & Hall/CRC, 2013..
[13] A.Y. Ng, M.I. Jordan, Y. Weiss, On spectral clustering: analysis and an algorithm, in: NIPS, 2001, pp. 849–856.
[14] L. Zelnik-Manor, Self-tuning spectral clustering, NIPS 17 (2004) 1601–1608.
[15] G. Hinton, R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.
[16] H. Lee, R. Grosse, R. Ranganath, A.Y. Ng, Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations, in: ICML, 2009, pp. 609–616..
[17] F. Schroff, D. Kalenichenko, J. Philbin, Facenet: A unified embedding for face recognition and clustering, in: CVPR, 2015, pp. 815–823.
[18] J. Bruna, S. Mallat, Invariant scattering convolution networks, TPAMI 35 (8) (2013) 1872–1886.
[19] J. Wang, J. Jiang, Sa-net: A deep spectral analysis network for image clustering, Neurocomputing 383 (2020) 10–23.
[20] J. Yang, D. Parikh, D. Batra, Joint unsupervised learning of deep representations and image clusters, in: CVPR, 2016..
[21] B. Yang, X. Fu, N.D. Sidiropoulos, M. Hong, Towards k-means-friendly spaces: Simultaneous deep learning and clustering, in: ICML, vol. 70, 2017, pp. 3861–3870..
[22] J. Xie, R. Girshick, A. Farhadi, Unsupervised deep embedding for clustering analysis, in: ICML, 2016, pp. 478–487.
[23] K.G. Dizaji, A. Herandi, H. Huang, Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization, in: ICCV, 2017, pp. 5747–5756.
[24] C.C. Hsu, C.W. Lin, Cnn-based joint clustering and representation learning with feature drift compensation for large-scale image data, TMM 20 (2) (2018) 421–429.
[25] U. Shaham, K.P. Stanton, H. Li, R. Basri, B. Nadler, Y. Kluger, Spectralnet: Spectral clustering using deep neural networks, in: ICLR, 2018.
[26] X. Yang, C. Deng, F. Zheng, J. Yan, W. Liu, Deep spectral clustering using dual autoencoder network, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.
[27] L. Yang, N.-M. Cheung, J. Li, J. Fang, Deep clustering by gaussian mixture variational autoencoders with graph embedding, in: The IEEE International Conference on Computer Vision (ICCV), 2019.
[28] A. Stuhlsatz, J. Lippel, T. Zielke, Feature extraction with deep neural networks by a generalized discriminant analysis, TNNLS 23 (2012) 596–608.
[29] Q. Wang, P. Li, L. Zhang, G^2denet: Global gaussian distribution embedding network and its application to visual recognition, in: CVPR, 2017.
[30] W. Zhang, D. Zhao, X. Wang, Agglomerative clustering via maximum incremental path integral, Pattern Recognition 46 (11) (2013) 3056–3065.
[31] S. Lloyd, Least squares quantization in pcm, IEEE Transactions on Information Theory 28 (2) (1982) 129–137.
[32] G. Celeux, G. Govaert, Gaussian parsimonious clustering models, Pattern Recognition 28 (5) (1995) 781–793.
[33] E. Elhamifar, R. Vidal, Sparse subspace clustering: algorithm, theory, and applications, IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (11) (2013) 2765–2781.
[34] C. You, D.P. Robinson, R. Vidal, Scalable sparse subspace clustering by orthogonal matching pursuit, in: CVPR, 2016, pp. 3918–3927.
[35] S. Huang, Z. Xu, Z. Kang, Y. Ren, Regularized nonnegative matrix factorization with adaptive local structure learning, Neurocomputing 382 (2020) 196–209.
[36] J. Pan, N. Gillis, Generalized separable nonnegative matrix factorization, IEEE Transactions on Pattern Analysis and Machine Intelligence (2019) 1.
[37] S. Huang, H. Wang, T. Li, T. Li, Z. Xu, Robust graph regularized nonnegative matrix factorization for clustering, Data Mining and Knowledge Discovery 32 (2) (2018) 483–503.
[38] X. Chen, D. Cai, Large scale spectral clustering with landmark-based representation, in: AAAI, 2011, pp. 313–318.
[39] D. Cai, X. He, X. Wang, H. Bao, J. Han, Locality preserving nonnegative matrix factorization, in: IJCAI, 2009.
[40] G. Trigeorgis, K. Bousmalis, S. Zafeiriou, B.W. Schuller, A deep semi-nmf model for learning hidden representations, in: ICML, 2014, pp. II-1692–II-1700..
[41] J. Wang, A. Hilton, J. Jiang, Spectral analysis network for deep representation learning and image clustering, in: ICME, 2019, pp. 1540–1545.
[42] C. Doersch, A. Gupta, A.A. Efros, Unsupervised visual representation learning by context prediction, in: ICCV, 2015..
[43] H.-Y. Lee, J.-B. Huang, M.K. Singh, M.-H. Yang, Unsupervised representation learning by sorting sequence, in: ICCV, 2017..
[44] K. Ghasedi, X. Wang, C. Deng, H. Huang, Balanced self-paced learning for generative adversarial clustering network, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.
[45] Y. Ren, N. Wang, M. Li, Z. Xu, Deep density-based image clustering, Knowledge-based Systems 197 (2020) 105841.
[46] X. Ji, J.F. Henriques, A. Vedaldi, Invariant information clustering for unsupervised image classification and segmentation, in: The IEEE International Conference on Computer Vision (ICCV), 2019.
[47] J. Wu, K. Long, F. Wang, C. Qian, C. Li, Z. Lin, H. Zha, Deep comprehensive correlation mining for image clustering, in: The IEEE International Conference on Computer Vision (ICCV), 2019.
[48] X. Guo, E. Zhu, X. Liu, J. Yin, Deep embedded clustering with data augmentation, in: Proceedings of The 10th Asian Conference on Machine Learning, ACML 2018, Beijing, China, November 14–16, 2018, 2018, pp. 550–565..
[49] Y. Ren, K. Hu, X. Dai, L. Pan, S.C.H. Hoi, Z. Xu, Semi-supervised deep embedded clustering, Neurocomputing 325 (2019) 121–130.
[50] H. Nakayama, T. Harada, Y. Kuniyoshi, Global gaussian approach for scene categorization using information geometry, in: CVPR, 2010.
[51] G. Serra, C. Grana, M. Manfredi, R. Cucchiara, Gold: Gaussians of local descriptors for image representation, CVIU 134 (2015) 22–32.
[52] K. Sohn, D.Y. Jung, H. Lee, A.O. Hero, Efficient learning of sparse, distributed, convolutional feature representations for object recognition, in: ICCV, 2011, pp. 2643–2650..
[53] L. Deng, J. Chen, Sequence classification using the high-level features extracted from deep neural networks, in: ICASSP, 2014, pp. 6844–6848.
[54] C.H. Lee, S.B. Hsu, J.L. Shih, C.H. Chou, Continuous birdsong recognition using gaussian mixture modeling of image shape features, TMM 15 (2) (2013) 454–464.
[55] E. Variani, E. Mcdermott, G. Heigold, A gaussian mixture model layer jointly optimized with discriminative features within a deep neural network architecture, in: ICASSP, 2015, pp. 4270–4274.
[56] G. Heigold, H. Ney, P. Lehnen, T. Gass, R. Schluter, Equivalence of generative and log-linear models, IEEE Transactions on Audio Speech and Language Processing 19 (5) (2011) 1138–1148.
[57] M. Paulik, Lattice-based training of bottleneck feature extraction neural networks, in: INTERSPEECH, 2013.

[58] Z. Tüske, M.A. Tahir, R. Schlüter, H. Ney, Integrating gaussian mixtures into deep neural networks: Softmax layer with hidden variables, in: ICASSP, 2015, pp. 4285–4289.
[59] G. Heigold, A log-linear discriminative modeling framework for speech recognition, Ph.D. dissertation, 2010..
[60] Z. Jiang, Y. Zheng, H. Tan, B. Tang, H. Zhou, Variational deep embedding: An unsupervised and generative approach to clustering, in: IJCAI, 2017, pp. 1965–1972.
[61] N. Dilokthanakul, P.A.M. Mediano, M. Garnelo, M.C.H. Lee, H. Salimbeni, K. Arulkumaran, M. Shanahan, Deep unsupervised clustering with gaussian mixture variational autoencoders, CoRR (2016).
[62] A. Graves, Stochastic backpropagation through mixture density distributions, CoRR (2016).
[63] A. van den Oord, B. Schrauwen, Factoring variations in natural images with deep gaussian mixture models, in: NIPS, 2014, pp. 3518–3526.
[64] J. Wang, J. Jiang, An unsupervised deep learning framework via integrated optimization of representation learning and gmm-based modeling, in: ACCV, 2018, pp. 249–265.
[65] X. Li, W. Hu, C. Shen, A. Dick, Z. Zhang, Context-aware hypergraph construction for robust spectral clustering, TKDE 26 (10) (2014) 2588–2597.
[66] D.A. Reynolds, Gaussian mixture models, in: Encyclopedia of Biometrics, 2009..
[67] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.
[68] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017..
[69] T. Sim, S. Baker, M. Bsat, The cmu pose, illumination, and expression (pie) database, in: IEEE International Conference on Automatic Face and Gesture Recognition Pittsburgh, 2002.
[70] S.A. Nene, S.K. Nayar, H. Murase, Columbia university image library (coil-20), 1996..
[71] S.A. Nene, S.K. Nayar, H. Murase, Columbia university image library (coil-100), 1996..
[72] A. Coates, A. Ng, H. Lee, An analysis of single-layer networks in unsupervised feature learning 15 (2011) 215–223..
[73] D.D. Lewis, Y. Yang, T.G. Rose, F. Li, Rcv1: A new benchmark collection for text categorization research, JMLR 5 (2004) 361–397.
[74] C. Doersch, S. Singh, A. Gupta, J. Sivic, A.A. Efros, What makes paris look like paris?, ACM Transactions on Graphics 31 (4) (2012) 101:1–101:9..
[75] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, JMLR 15 (2014) 1929–1958.
[76] D. Peel, G.J. McLachlan, Robust mixture modelling using the t distribution, Statistics and Computing 10 (4) (2000) 339–348.
[77] A.K. Jain, M.N. Murty, P.J. Flynn, Data clustering: a review, ACM Computing Surveys 31 (3) (1999) 264–323.
[78] D. Zhao, X. Tang, Cyclizing clusters via zeta function of a graph, in: NIPS, 2009, pp. 1953–1960.
[79] W. Zhang, X. Wang, D. Zhao, X. Tang, Graph degree linkage: Agglomerative clustering on a directed graph, in: ECCV, 2012, pp. 428–441.
[80] W. Xu, X. Liu, Y. Gong, Document clustering based on non-negative matrix factorization, in: ACM SIGIR, 2003, pp. 267–273.
[81] D. Cai, X. He, J. Han, Locally consistent concept factorization for document clustering, TKDE 23 (6) (2011) 902–913.
[82] P. Vincent, H. Larochelle, Y. Bengio, P. Manzagol, Extracting and composing robust features with denoising autoencoders, in: ICML, 2008, pp. 1096–1103..
[83] L. Maaten, Learning a parametric embedding by preserving local structure, in: the 12nd International Conference on Artificial Intelligence and Statistics, vol. 5, 2009, pp. 384–391..

Jinghua Wang received his Ph.D. degree from The Hong Kong Polytechnic University in 2013. From 2014 to 2016, he was a research fellow with Nanyang technological University. He is currently a Research Assistant Professor with Shenzhen University. His current research interests include computer vision and machine learning.

Jianmin Jiang received PhD from the University of Nottingham, UK, in 1994. From 1997 to 2001, he worked as a full professor of Computing at the University of Glamorgan, Wales, UK. In 2002, he joined the University of Bradford, UK, as a Chair Professor of Digital Media, and Director of Digital Media & Systems Research Institute. He worked at the University of Surrey, UK, as a full professor during 2010–2014 and a distinguished professor (1000-plan) at Tianjin University, China, during 2010–2013. He is currently a Distinguished Professor and director of the Research Institute for Future Media Computing at the College of Computer Science & Software Engineering, Shenzhen University, China. He was a chartered engineer, fellow of IEE, fellow of RSA, member of EPSRC College in the UK, and EU FP-6/7 evaluator. His research interests include, image/video processing in compressed domain, digital video coding, medical imaging, computer graphics, machine learning and AI applications in digital media processing, retrieval and analysis. He has published around 400 refereed research papers.