

Deep Clustering by Gaussian Mixture Variational Autoencoders with Graph Embedding

Linxiao Yang^{1,2}, Ngai-Man Cheung^{†1}, Jiaying Li¹, and Jun Fang²

¹Singapore University of Technology and Design (SUTD)

²University of Electronic Science and Technology of China

[†]Corresponding author: ngai_man_cheung@sutd.edu.sg

Abstract

We propose *DGG*: Deep clustering via a Gaussian-mixture variational autoencoder (VAE) with Graph embedding. To facilitate clustering, we apply Gaussian mixture model (GMM) as the prior in VAE. To handle data with complex spread, we apply graph embedding. Our idea is that graph information which captures local data structures is an excellent complement to deep GMM. Combining them facilitates the network to learn powerful representations that follow global model and local structural constraints. Therefore, our method unifies model-based and similarity-based approaches for clustering. To combine graph embedding with probabilistic deep GMM, we propose a novel stochastic extension of graph embedding: we treat samples as nodes on a graph and minimize the weighted distance between their posterior distributions. We apply Jensen-Shannon divergence as the distance. We combine the divergence minimization with the log-likelihood maximization of the deep GMM. We derive formulations to obtain an unified objective that enables simultaneous deep representation learning and clustering. Our experimental results show that our proposed *DGG* outperforms recent deep Gaussian mixture methods (model-based) and deep spectral clustering (similarity-based). Our results highlight advantages of combining model-based and similarity-based clustering as proposed in this work. Our code is published here: <https://github.com/dodoyang0929/DGG>.

1. Introduction

Clustering aims to classify data into several classes without label information [15]. It is one of the fundamental

tasks of unsupervised learning. A number of methods have been proposed [38, 19, 8]. Based on the approaches to model the space structure, most clustering methods can be classified into two categories, namely, *model based methods* and *similarity based methods*. The model based methods, such as the Gaussian mixture model [4] and subspace clustering [1, 36], focus on the global structure of the data space. They put assumptions on the whole data space and fit the data using some specific models. An advantage of model based methods is their good generalization ability. Once trained, new samples can be readily clustered using the learnt model parameters. However, it is challenging for these methods to deal with data with complex spread. Different from model based methods, the similarity based methods emphasize the local structure of the data. These methods formulate the local structures using some similarities or distances between the samples. Spectral clustering [33, 26], a popular similarity-based method, constructs a graph using the sample similarities, and treats the smoothest signals on the graph as the features of the data. With mild assumption, similarity-based methods achieve tremendous success [25]. Many similarity-based methods, however, suffer from high computational complexity. Spectral clustering, for instance, requires to perform a singular value decomposition when computing features, which is prohibitive for large datasets. To address this issue, a lot of efforts have been made and many methods have been proposed [5, 10, 22, 39].

Deep clustering Recent advanced deep learning technique offers new opportunities for clustering [24]. With powerful capability to learn non-linear mapping, deep learning provides a promising feature learning framework [41, 37, 42]. Several works have considered to combine the model-based clustering approach with deep learning, where global assumptions were imposed on the feature space [17, 7]. These methods jointly train the network to

Work done at SUTD

learn better features, and use the clustering results to direct the network training. Leveraging the excellent feature learning ability of deep neural networks, these methods substantially outperform traditional clustering methods. Several methods [21, 31] have also been proposed to combine deep learning with similarity-based clustering method to address the limitations of generalization and scalability. SpectralNet [31], for example, is a recent work to learn a mapping that maps data to their spectral embedding using a deep neural network. The network is trained using the mini-batch stochastic gradient descent to make sure that the method is applicable for large scale dataset. Meanwhile, once trained, embeddings of new samples can be obtained via forward pass of the network. Therefore, the model is easy to generalize. However, the method still needs to perform a Cholesky factorization in each epoch.

Our contribution. In spite of the success of existing clustering methods, few of them consider to combine the model-based and similarity-based approaches. As the model-based and similarity-based approaches focus on the global and local structures of the data respectively, their combination can potentially lead to powerful representation for clustering. With this motivation, in this work, we propose a framework to combine model based and similarity based approaches. As will be discussed, our work can be viewed as an improvement of a model-based method with local structure constraint to handle data with complex spread. Alternatively, our work can also be viewed as an improvement of a similarity based method with imposing a global model explicitly in the latent space. Our model is based on variational autoencoder [18], and we set the prior distribution of the latent features to a Gaussian mixture distribution. To combine graph embedding with this probabilistic deep GMM, we propose a *stochastic* extension of graph embedding: We treat the data as the nodes of a sample similarity graph, and minimize the weighted distance between their posterior distributions to exploit the similarity information. We propose to use the Jensen-Shannon (JS) divergence as the distance. Then, we relax it to its variational upper bound. After some formulation, we combine the loss of stochastic graph embedding with the objective function of deep GMM, and eventually derive a unified loss function, which can be optimized using established reparameterization trick and gradient descent. To illustrate the superior of the proposed method, we perform experiments on synthetic data and real-world data.

2. Related Work

The main idea of the recent deep learning-based clustering methods is to learn latent features of the training data using deep neural networks. Then, traditional clustering methods are applied to compute the cluster assignments. To preserve some structure in the feature space, additional con-

straints are usually included in the network. Specifically, DCN [41] incorporates clustering loss in the loss function of an autoencoder to jointly refine the latent features and cluster assignments. DEC [37] trains an autoencoder to learn features and imposes a soft assignment constraint on the them. DEPICT [7] consists of a convolutional autoencoder and a single layer classifier, which learns the latent features and the distribution of the cluster assignments, respectively. The neural network is optimized by minimizing the reconstruction error and the relative entropy between the distributions of the cluster assignments and their prior. To make the network more robust, clear and noisy images are trained jointly. IMSAT [14] learns a discrete latent feature using a network by maximizing the mutual information between the training images and their latent features. To further improve performance, IMAST augments the dataset by adding permutation of training samples, and assumes that their latent features produced by the network are invariant. SpectralNet [31] is a method based on the spectral clustering idea. It attempts to learn a network that maps the training data into the eigenspace of the graph Laplacian matrix. They apply a Siamese network to learn the weights between the graph nodes, and use k-means to perform the final clustering. Apart from these methods, several generative model based methods have also been proposed. Most of them are based on the variational autoencoder framework, but replace the prior of the latent variable to some specific distributions, such as mixture of Gaussian distributions. Specifically, VaDE [17] and GMVAE [7] assume the latent variables follows a mixture of Gaussian, where the means and variances of the Gaussian components are trainable. The LTVAE[23] assumes the latent variables obey a tree structure model, and iteratively update the structure to capture the facets of the data.

The work that is mostly related to our work is VaDE. Both VaDE and our work learn a Gaussian mixture model for the latent features. However, the difference is clear and significant: our method applies graph embedding to preserve local structures of the data. Specifically, in VaDE, the distribution of the latent features are learnt independently. On the other hand, in our proposed method, we push the latent distributions of the training samples to become close to each other if the training samples are connected on the sample similarity graph. With the assistance of the graph, our proposed method is able to learn powerful representations and handle data with complex spread. Experiment results suggest our proposed method outperforms VaDE.

3. Proposed Method

In this section, we first discuss the deep GMM as the base of of method. Then, we discuss our main contribution: combining graph embedding in the deep generative model. We discuss our proposed formulation to combine stochas-

tic graph embedding regularization with the loss of deep GMM. We also discuss update of parameters and construction of the affinity matrix of the graph.

3.1. Deep Gaussian Mixture Model

Given a set of D -dimensional training samples $\{\mathbf{x}_n\}_{n=1}^N$, we aim to cluster them into K classes. For each training sample \mathbf{x} , we learn a latent feature $\mathbf{z} \in \mathbb{R}^{M \times 1}$ for it. We assume the latent features follows a Gaussian mixture distribution. We introduce a binary vector $\mathbf{c} \in \{0, 1\}^{K \times 1}$ to indicate which Gaussian component that the latent feature \mathbf{z} belongs to.

3.1.1 Generative model

In our model, we assume the data is drawn from a Gaussian mixture distribution. Specifically, for a sample \mathbf{x} , we model its generative process as follows:

$$p(\mathbf{c}; \boldsymbol{\theta}) = \prod_{k=1}^K c_k^{\mu_k} \quad (1)$$

$$p(\mathbf{z} | c_k = 1) = \mathcal{N}(\boldsymbol{\mu}_k, \text{diag}(\boldsymbol{\sigma}_k^2)) \quad (2)$$

$$p(\mathbf{x} | \mathbf{z}) = \begin{cases} \text{Ber}(\boldsymbol{\mu}_x) & \text{if } \mathbf{x} \text{ is binary} \\ \mathcal{N}(\boldsymbol{\mu}_x, \mathbf{I}) & \mathbf{x} \text{ is real-valued} \end{cases} \quad (3)$$

where c_k and μ_k denote the k th entry of \mathbf{c} and $\boldsymbol{\mu}$, respectively, μ_k satisfied $\sum_{k=1}^K \mu_k = 1$, $\boldsymbol{\mu}_k$ and $\boldsymbol{\sigma}_k^2$ denotes the mean and variance of the k th Gaussian component, respectively, $\boldsymbol{\mu}_x = g(\mathbf{z}; \boldsymbol{\theta})$, \mathbf{I} denotes the identity matrix, $\boldsymbol{\theta}$ is a predefined parameter, and g is a neural network with trainable parameters.

3.1.2 Inference model

Directly solving the generative model, i.e. finding maximum a posterior (MAP) of latent variables and maximum likelihood estimation (MLE) of parameters, is difficult. To address this issue, we approximate the posterior distribution $p(\mathbf{z}, \mathbf{c} | \mathbf{x})$ using a new distribution $q(\mathbf{z}, \mathbf{c} | \mathbf{x})$, which is drawn from specific class and parameterized by trainable parameter $\boldsymbol{\theta}$. Specifically, we assume $q(\mathbf{z}, \mathbf{c} | \mathbf{x})$ can be factorized as $q(\mathbf{z}, \mathbf{c} | \mathbf{x}) = q_1(\mathbf{z} | \mathbf{x})q_2(\mathbf{c} | \mathbf{z})$. Then we define

$$q_1(\mathbf{z} | \mathbf{x}) = \mathcal{N}(\bar{\boldsymbol{\mu}}, \text{diag}(\bar{\boldsymbol{\sigma}}^2)) \quad (4)$$

$$q_2(\mathbf{c} | \mathbf{z}) = \text{Multinomial}(\bar{\boldsymbol{\theta}}) \quad (5)$$

where

$$[\bar{\boldsymbol{\mu}}, \log(\bar{\boldsymbol{\sigma}}^2)] = \mathbf{f}_1(\mathbf{x}; \boldsymbol{\theta}_1) \quad (6)$$

$$\bar{\boldsymbol{\theta}} = \mathbf{f}_2(\mathbf{z}; \boldsymbol{\theta}_2) \quad (7)$$

Here \mathbf{f}_1 and \mathbf{f}_2 denote neural networks with parameters $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$, respectively. The model contains two neural networks. The first one \mathbf{f}_1 learns the latent distributions from

the training samples, and the second one \mathbf{f}_2 computes the probability of which Gaussian component the latent features belong to.

With the framework of the generative model and inference model, the parameters can be estimated by maximizing the log-likelihood function, i.e.,

$$\max_{\boldsymbol{\theta}} \sum_{i=1}^N \ln p(\mathbf{x}_i) \quad (8)$$

The problem (8) is usually solved by maximizing the evidence lower bound (ELBO) of the log-likelihood function with reparameterization trick [18]. We note that the proposed model consists three networks, i.e., g , \mathbf{f}_1 , and \mathbf{f}_2 . We refer g and \mathbf{f}_1 as the decoder and encoder, respectively, as they form an variational autoencoder (VAE). We refer \mathbf{f}_2 as the classifier, as it classifies the latent features into one of the K classes.

3.2. VAE with Graph Embedding

Graph embedding [40, 28, 2, 13, 16, 29, 27] aims to find the low-dimension features that preserve the similarity relationship between the vertex pairs in a sample similarity graph. Generally, under graph embedding, training samples $\{\mathbf{x}_n\}$ are viewed as the vertexes of a similarity graph that is characterized by an affinity matrix \mathbf{W} . The optimal features $\{\mathbf{z}_n\}$ are found by [40]

$$\{\mathbf{z}_n\} = \arg \min_{\mathbf{Z} \mathbf{Z}^T = \mathbf{I}} \sum_{i=1}^N \sum_{j=1}^N w_{ij} \|\mathbf{z}_i - \mathbf{z}_j\|^2 \quad (9)$$

where $\mathbf{Z} = [\mathbf{z}_1, \dots, \mathbf{z}_N]$, w_{ij} denotes the (i, j) th element of \mathbf{W} . The constraint $\mathbf{Z} \mathbf{Z}^T = \mathbf{I}$ is used to avoid trivial solutions. From (9), we can see that if samples are connected on the graph, their features will be close to each other. This inspires us that if the two samples are connected on the graph, they should have similar latent features and cluster assignments. As we have discussed, in our model, the latent features and cluster assignments are random variables. Therefore, we propose to measure the distance of the posterior distributions. To this end, we propose to add a constraint to the problem (8), and arrive at

$$\max_{\boldsymbol{\theta}} \sum_{i=1}^N \ln p(\mathbf{x}_i) - \sum_{j=1}^N w_{ij} d(q(\mathbf{z}, \mathbf{c} | \mathbf{x}_i), q(\mathbf{z}, \mathbf{c} | \mathbf{x}_j)) \quad (10)$$

where $d(\cdot, \cdot)$ is a metric that measure the distance between two distributions. To balance the weight of each training samples, we further require $\sum_j w_{ij} = 1$. Note that (10) is similar to (9) in spirit, but formulate the relationship between the features stochastically. Moreover, we impose the

graph embedding constraint not only on the features, but also on the cluster assignments. This makes the clustering results smooth w.r.t. the graph. Note that, different from traditional methods that need additional constraint, our model with an autoencoder can automatically avoid trivial solutions by forcing the features able to reconstruct the original data.

The selection of $d(\cdot, \cdot)$ plays a critical role for the proposed method. We note that it is not trivial to select a proper $d(\cdot, \cdot)$: the latent feature z and the cluster assignment c are related, and there is no analytical expression for their joint distribution, which makes most divergence difficult to apply. In this paper, we select the Jensen-Shannon (JS) divergence [9], i.e.,

$$\max_{\theta} \sum_{i=1}^N \ln p(\mathbf{x}_i) - \sum_{j=1}^N w_{ij} \text{JS}(q(z, c|\mathbf{x}_i), q(z, c|\mathbf{x}_j)) \quad (11)$$

We will show that after appropriate relaxation, (11) can be solved.

3.2.1 Learning algorithms

In this subsection, we discuss how to solve (11). It is noted that there is no closed form solution of the JS divergence between $q(z, c|\mathbf{x}_i)$ and $q(z, c|\mathbf{x}_j)$. Thus minimizing the JS divergence has to resort to the reparameterization trick, which will lead to large estimate variance, as the posterior $q(z, c|\mathbf{x})$ is related to all the K Gaussian component. To overcome the difficulty, instead of directly minimizing the JS divergence, we minimize a variational upper bound of it. We define

$$G(\theta, \mathbf{x}_i, \mathbf{x}_j) = \frac{1}{2} \text{KL}(q(z, c|\mathbf{x}_i) \| p(z, c|\mathbf{x}_i)) + \frac{1}{2} \text{KL}(q(z, c|\mathbf{x}_j) \| p(z, c|\mathbf{x}_j)) \quad (12)$$

and notice

$$\begin{aligned} G(\theta, \mathbf{x}_i, \mathbf{x}_j) &= \text{JS}(q(z, c|\mathbf{x}_i) \| q(z, c|\mathbf{x}_j)) + \text{KL}(M \| p(z, c|\mathbf{x}_i)) \\ &\quad \text{JS}(q(z, c|\mathbf{x}_i) \| q(z, c|\mathbf{x}_j)) \end{aligned} \quad (13)$$

where $M = \frac{1}{2}(q(z, c|\mathbf{x}_i) + q(z, c|\mathbf{x}_j))$, and the equation holds when $M = p(z, c|\mathbf{x}_i)$. We arrive at that $G(\theta, \mathbf{x}_i, \mathbf{x}_j)$ is an upper bound of the JS divergence between $q(z, c|\mathbf{x}_i)$ and $q(z, c|\mathbf{x}_j)$. Thus we can minimize the upper bound $G(\theta, \mathbf{x}_i, \mathbf{x}_j)$ with respect to θ and alternatively to downhill the JS divergence between the posteriors. Specifically, we first minimize $G(\theta, \mathbf{x}_i, \mathbf{x}_j)$ with respect to θ to reduce the gap between the JS divergence and its upper bound. Then we fixed θ and minimize the

$G(\theta, \mathbf{x}_i, \mathbf{x}_j)$ with respect to θ to decrease the value of JS divergence.

In the following, we show that $G(\theta, \mathbf{x}_i, \mathbf{x}_j)$ can be absorbed in the log-likelihood function. For a training sample \mathbf{x}_i , its log-likelihood $\ln p(\mathbf{x}_i)$ can be decomposed as [4]

$$\begin{aligned} \ln p(\mathbf{x}_i) &= \text{KL}(q(z, c|\mathbf{x}_i) \| p(z, c|\mathbf{x}_i)) \\ &\quad + E_{q(z, c|\mathbf{x}_i)} \ln \frac{p(\mathbf{x}_i, z, c)}{q(z, c|\mathbf{x}_i)} \end{aligned} \quad (14)$$

Surprisingly, the above equation still holds if we replace $q(z, c|\mathbf{x}_i)$ with $q(z, c|\mathbf{x}_j)$ (see derivation in supplementary), that is,

$$\begin{aligned} \ln p(\mathbf{x}_i) &= \text{KL}(q(z, c|\mathbf{x}_j) \| p(z, c|\mathbf{x}_i)) \\ &\quad + E_{q(z, c|\mathbf{x}_j)} \ln \frac{p(\mathbf{x}_i, z, c)}{q(z, c|\mathbf{x}_j)} \end{aligned} \quad (15)$$

Averaging the equation (14) and (15), we have

$$\begin{aligned} \ln p(\mathbf{x}_i) &= G(\theta, \mathbf{x}_i, \mathbf{x}_j) \\ &\quad + \frac{1}{2} (\text{L}(\theta, \mathbf{x}_i) + \text{L}(\theta, \mathbf{x}_i, \mathbf{x}_j)) \end{aligned} \quad (16)$$

where

$$\text{L}(\theta, \mathbf{x}_i) = E_{q(z, c|\mathbf{x}_i)} \ln \frac{p(\mathbf{x}_i, z, c)}{q(z, c|\mathbf{x}_i)} \quad (17)$$

$$\text{L}(\theta, \mathbf{x}_i, \mathbf{x}_j) = E_{q(z, c|\mathbf{x}_j)} \ln \frac{p(\mathbf{x}_i, z, c)}{q(z, c|\mathbf{x}_j)} \quad (18)$$

Recall the constraint that $\sum_j w_{ij} = 1$, we can rewrite the objective function as

$$\max_{\theta} \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N w_{ij} (\text{L}(\theta, \mathbf{x}_i) + \text{L}(\theta, \mathbf{x}_i, \mathbf{x}_j)) \quad (19)$$

An intuitive explanation of the objective function (19) is that: for a sample \mathbf{x}_i , it should not only be reconstructed by the feature of itself, but also be reconstructed by that of \mathbf{x}_j . This corroborates the success of [6, 30, 32], where a similar strategy is used, but none of them provides explanation.

We now evaluate the objective function in (19) according to the proposed inference and generative model. Recall the dependency of the variables, $\text{L}(\theta, \mathbf{x}_i)$ can be rewritten as

$$\begin{aligned} \text{L}(\theta, \mathbf{x}_i) &= E_{q(z|\mathbf{x}_i)q(c|z)} \ln \frac{p(\mathbf{x}_i|z)p(z|c)p(c)}{q(z|\mathbf{x}_i)q(c|z)} \end{aligned} \quad (20)$$

Substituting (4), (5), (1), (2), (3) into (20), and applying the

reparameterization trick, $L(\theta, \phi; \mathbf{x}_i)$ can be given as

$$L(\theta, \phi; \mathbf{x}_i) = \sum_{d=1}^D x_d^i \log \mu_{x_i|d} + (1 - x_d^i) \log(1 - \mu_{x_i|d}) - \sum_{k=1}^K \sum_{m=1}^M (\log \frac{2}{k|m} + \frac{\tilde{z}_i^2|m}{2k|m} + \frac{(\tilde{\mu}_i|m - \mu_{k|m})^2}{2k|m}) + \sum_{k=1}^K \sum_{m=1}^M \log \frac{ik}{jk} + \frac{1}{2} \sum_{m=1}^M (1 + \log \frac{\tilde{z}_i^2|m}{2}) \quad (21)$$

where D is the dimension of \mathbf{x} and μ_{x_i} , x_d^i is the d th element of \mathbf{x}_i , and $|_m$ denotes the m th element of $\tilde{\mathbf{z}}$, ik denotes $q(c_k = 1|z_i)$ for simplicity. Here

$$\mu_{x_i} = g(\mathbf{z}_i; \theta) \quad (22)$$

$$\mathbf{z}_i = \tilde{\mu}_i + \tilde{\epsilon}_i \quad (23)$$

$$[\tilde{\mu}_i, \log(\tilde{\epsilon}_i^2)] = f(\mathbf{x}_i, \phi) \quad (24)$$

where $\tilde{\epsilon}_i$ is a vector with all its entries are drawn independently from the normal distribution, and “ \odot ” denotes element-wise product.

Similarly, $L(\theta, \phi; \mathbf{x}_i, \mathbf{x}_j)$ can be rewritten as

$$L(\theta, \phi; \mathbf{x}_i, \mathbf{x}_j) = E_{q(z_i|x_j)q(c|z)} \ln \frac{p(\mathbf{x}_i|z)p(z|c)p(c)}{q(z|\mathbf{x}_j)q(c|z)} \quad (25)$$

and be finally evaluated by

$$L(\theta, \phi; \mathbf{x}_i, \mathbf{x}_j) = \sum_{d=1}^D x_d^i \log \mu_{x_j|d} + (1 - x_d^i) \log(1 - \mu_{x_j|d}) - \sum_{k=1}^K \sum_{m=1}^M (\log \frac{2}{k|m} + \frac{\tilde{z}_j^2|m}{2k|m} + \frac{(\tilde{\mu}_j|m - \mu_{k|m})^2}{2k|m}) + \sum_{k=1}^K \sum_{m=1}^M \log \frac{ik}{jk} + \frac{1}{2} \sum_{m=1}^M (1 + \log \frac{\tilde{z}_j^2|m}{2}) \quad (26)$$

where

$$\mu_{x_j} = g(\mathbf{z}_j; \theta) \quad (27)$$

$$\mathbf{z}_j = \tilde{\mu}_j + \tilde{\epsilon}_j \quad (28)$$

$$[\tilde{\mu}_j, \log(\tilde{\epsilon}_j^2)] = f(\mathbf{x}_j, \phi) \quad (29)$$

3.2.2 Update of Parameters

We discuss the update of the parameters in our model. The parameters $\{\mu_{k|}, \sigma_{k|}\}_{k=1}^K$ are updated using the mini-batch stochastic gradient descent as they are related to all the

training samples. For the parameters $\{\epsilon_i\}_{i=1}^N$, we optimize them by maximizing (19). In what follows, we discuss the update for ϵ_i . Substitute (21) and (26) into (19), and remove the terms independent with ϵ_i , then the optimal ϵ_i is given as

$$\epsilon_i = \underset{\epsilon_i}{\operatorname{argmax}} \sum_{k=1}^K w_{ij} (\sum_{k=1}^K (\epsilon_{ik} + \epsilon_{jk}) \ln \epsilon_{ik}) \quad (30)$$

where ϵ_{ik} denotes the k th entry of ϵ_i , and ϵ_i denotes the set $\{\epsilon_j | w_{ij} = 0\}$. Using the Lagrange multiplier method [3], ϵ_{ik} can be updated as

$$\epsilon_{ik} = \frac{\sum_j w_{ij} \epsilon_{jk}}{\sum_j w_{ij} (\epsilon_{ik} + \epsilon_{jk})} \quad (31)$$

3.3 Construction of the Affinity Matrix

Similar to other graph embedding methods, it is important to have a properly constructed affinity matrix. A typical selection of the affinity matrix is to find a set nearest neighbors for a given data point and compute their similarity using a predefined kernel function. For instance, with Gaussian kernel, the element of the affinity matrix is defined as

$$w_{ij} = \begin{cases} \frac{1}{a_i} \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2s_i^2}) & \text{if } \mathbf{x}_j \in N(\mathbf{x}_i) \\ 0, & \text{otherwise} \end{cases} \quad (32)$$

where s_i is a predefined scalar, $N(\mathbf{x}_i)$ denotes the set consist of the nearest N_s neighbors of \mathbf{x}_i , and a_i is a normalizer that makes $\sum_j w_{ij} = 1$.

To make the proposed method more robust to the different data set, we train a Siamese network [11, 31] to measure the similarity between the data points. The details of the Siamese network can be found in the Appendix.

In summary, we have introduced a deep graph embedding-based Gaussian mixture VAE for clustering. We summarize our proposed method in Algorithm 1.

4 Experiments

We conduct experiments to show the superior of the proposed method. Throughout our experiments, we initialize the network with the following procedure. We first perform greedy layer-wise training in denoising autoencoder manner, and then stack them into a deep autoencoder. Then, the network is trained as a variational autoencoder. After we have pretrained the network, we collect the representations learnt by the pretrained network. K-means is employed to cluster the representations and generate pseudo-label, with which we train the classifier network f_2 . The means of the Gaussian mixture model is initialized using the cluster centers determined by K-means, and the variance is initialized by its unbiased estimator.

Figure 1. Results of the proposed method and VaDE on 2D examples with different cluster distance. From left to right: training data in Euclidean space, learnt latent features by VaDE, clustering result of VaDE, learnt latent features by DGG (our proposed method), clustering result of DGG. Compared to VaDE (model-based), improvement with combining model-based and similarity-based as in our DGG is clear.

Algorithm 1

Input: Training samples $\{\mathbf{x}_n\}_{n=1}^N$, number of desired clusters K , batch-size m .

Output: cluster index $\{c_n\}_{n=1}^N$ and parameters of GMM $\{\mu_k, \sigma_k\}_{k=1}^K$.

- 1: Build the directed graph according to Euclidean distance or Siamese distance, compute the affinity matrix and form the training tuples;
 - 2: **while** not converge **do**
 - 3: Draw a mini-batch training tuples $\{\mathbf{x}_i, \{\mathbf{x}_j\}_{j=1}^{m-1}\}$;
 - 4: Compute $q(\mathbf{z}|\mathbf{x}_i)$ and $\{q(\mathbf{z}|\mathbf{x}_j)\}_{j=1}^{m-1}$;
 - 5: Generate samples \mathbf{z}_i and $\{\mathbf{z}_j\}_{j=1}^{m-1}$ according to (23) and (28), respectively.
 - 6: Evaluate the objective function of (21) and (26) using the generated samples.
 - 7: Update network parameters and $\{\mu_k, \sigma_k\}_{k=1}^K$;
 - 8: Update μ_i via (31);
 - 9: **end while**
-

4.1. Synthetic data

We first show the effectiveness of the proposed method on the synthetic data. We generate two classes of 2-dimensional training samples. Each class contains 2000 points, and forms a half circle in the data space. We cluster these points using the proposed method and VaDE. For the proposed method, we use the network structures of $2 - 20 - 20 - 2$ for encoder, and $2 - 20 - 20 - 2$ for decoder, and $2 - 2$ for classifier. All the layers are fully connected. The activate function is ReLU. We select 40 nearest neighbors to construct the affine matrix according to (32). We use the Adam optimizer with initial learning rate set to 0.02 and decay every 10 epoch by a factor 0.9. Note that, for fair comparison, the network architecture and the training setup of VaDE are same as our proposed method. Fig.1 shows the cluster results and the learnt latent features (mean of

$q(\mathbf{z}|\mathbf{x})$) of respective methods on the samples with different inter-class distances. From Fig.1, we can observe that our proposed method produces promising cluster results for both cases with different class distances. On the other hand, VaDE cannot cluster the two classes correctly. With the assistance of graph information, our proposed method performs well even the two clusters are quite near.

Another observation from Fig.1 is that the latent features learnt by the proposed method are aligned with the coordinate. This is because, by the network design, the posterior distributions of the latent features are forced to have diagonal covariance matrices. The JS divergence decreases if two latent distributions (Gaussian distributions) align with the coordinate. This property can help learn disentangled representations for the training samples.

4.2. Real-world data

We conduct experiments on the real-world data to evaluate the proposed method. Several widely known datasets are used, namely, MNIST, STL-10, Reuters, and HHAR, where MNIST and STL-10 are image datasets, Reuters is a dataset consists of the TF-IDF features of the word, and HHAR is a sensor signal dataset. We vectorize the images in MNIST, and subtract features for STL-10 using the pretrained ResNet-50 [12]. After the preprocessing, the MNIST contains 10 classes of 786-dimensional training samples and 7000 samples for each class, the STL-10 contains 10 classes of 2048-dimensional training samples and 1300 samples for each class, the Reuters contains 4 classes of 2000-dimensional training samples and 10000 samples in total, and the HHAR contains 10 classes of 561-dimensional training samples and 10200 samples in total. We compare our method with several state-of-art deep learning based clustering methods, including AE+GMM, DEC, IMSAT, VaDE, SpectralNet, and LTVAE. For the proposed method, we use the network structure of D-500-500-

Table 1. Clustering accuracy of the respective methods

Method	MNIST	STL-10	Reuters	HHAR
AE+GMM	82.18	79.83	70.98	77.67
DEC [37]	84.3	80.64	74.32	79.86
IMSAT [14]	98.4 ± 0.4	94.1 ± 0.4	71.0 ± 4.9	-
VaDE [17]	94.46	84.45	79.83	84.46
SpectralNet [31]	97.1±0.1	-	80.3±0.6	-
LTVAE [23]	86.30	90.00	80.96	85.00
DGG (Proposed)	97.58±0.1	90.59±0.2	82.3±1.2	89.04±0.1

Figure 2. Images generated by the proposed model and estimated variance of the components in GMM. All the four sub-figure is generated similarly. We take sub-figure at the left upper as an example. Left part: images generated by sampling the latent code from one Gaussian component of the learnt GMM. The image at i th row j th column is generated by inputting $\mu + 4a_j(e_i)$ into the decoder, where μ and a_j are the mean and standard deviation of the learnt Gaussian component, e_i is a vector of length 10 with all of its elements equal to 0, but the i th one equals to 1, $a_j = -1 + (j - 1)/7$. Right part: from top to bottom: the bar at the i th row denotes the amplitude of the i th element of μ .

2000-10 for encoder, 10-2000-500-500-D for decoder, 10-L ($L = 10$) or 10-L-L ($L < 10$) for classifier, where D denotes the dimension of the training samples, and L denotes the number of classes. All layer are fully connected and ReLU is used as activate function. We randomly select 20 among 100 nearest neighbors generated by the Siamese network to construct affine matrix using (32). Adam optimizer is used with initial learning rate set to 0.02 and decays every 10 epochs with factor 0.9. The parameter η is set to 10, 10, 0.01 for STL-10, Reuters, and HHAR, respectively. We note that the network architecture used in the proposed method are same with that of VaDE and LTVAE for a fair comparison. We measure the performance of respective methods using the cluster accuracy, which is

defined as

$$\text{ACC} = \max_m \frac{\sum_n \mathbb{1}\{l_n = m(c_n)\}}{N} \quad (33)$$

where l_n and c_n denotes the ground-truth label and the cluster assignment generated by the algorithm for sample x_n , respectively. The m tries all the possible one-to-one mappings between the label and cluster.

We show the results of clustering accuracy achieved by the respective methods on Tab.1, and highlight the top two accuracy scores. From the Tab.1 we have following observations. 1) For MNIST and STL-10 datasets the IMSAT performs best, while the proposed method produce competitive clustering accuracy. For the Reuters dataset,

the proposed method achieve the highest clustering accuracy and outperform the IMAST by a large margin. 2) The proposed method substantially outperform VaDE. This supports our claim that although both the proposed method and VaDE are based on the Gaussian mixture model framework, the proposed method, however, also exploits the additional graph information and thus outperforms VaDE. 3) The proposed method outperforms SpectralNet, which also utilizes graph information. This is because SpectralNet is a two-stage method which first learns the latent features using the network with the affinity information and then performs clustering use k-means. Our proposed method, however, jointly learns the latent features and performs clustering using GMM, which make it superior than SpectralNet.

4.3. Generating Samples

Another advantage of the proposed method is that it can naturally be used to generate realistic images. More surprisingly, the latent features learnt by the proposed method, as analysed above, are tending to align with the coordinates, which leads to the variance of some coordinates of the Gaussian component collapse to zero. This is due to that the covariance matrix of the Gaussian components in GMM are forced to be diagonal, thus elements of the variance Σ_k capture the spread width of the latent features on corresponding coordinates. As the latent features are constrained to be able to reconstruct the original samples, the coordinates with small variance carry few information of the cluster, while these coordinates with large variance capture the variation tendency of the images in the cluster. This provides opportunity to estimate the numbers of the factors that control an image using the variance of the Gaussian component. To this end, we train our model on the MNIST dataset, and generate samples using the learnt Gaussian components. We plot the images generated by the decoder of our model as well as the learnt variance of the Gaussian components on Fig.2. From Fig.2, we see that the variance vector Σ_k of the learnt Gaussian are sparse or approximately sparse, which collaborates our claim that the learnt features align with the coordinates. From the images in the left-up corner of the Fig.2, we see that for digital number “1”, only two factors affect the image, i.e., the degree of the thinness and rotation, and the corresponding coordinates of the Gaussian component have large variance. The same factors also affect the number “0”, “7”, and “8” through the same coordinates. But apart from the degree of the thinness and rotation, the variances of the Gaussian component also reflect other factors that control the images of these digital number, such as width and height of the number. Moreover, the model is also able to identify some specific factors for the images, such as degree of the sharpness of the corner of the number “7”, the ratio between the sizes of the upper and lower circle of the number “8”. This ability is helpful when learn a

Table 2. Clustering accuracy with different number of neighbors

N_s	0	1	3	10	20	30
ACC	94.82	96.98	97.33	97.52	97.58	97.49

disentangled representation.

4.4. Impact of the number of neighbors

We further investigate the impact of the number of neighbors used to construct the affinity matrix. The affinity matrix is critical in our model. More neighbors will involve additional information helping the clustering, but it also increase the probability that including inconsistent neighbors, which may mislead the clustering. Tab.2 shows the average performance of the proposed method on MNIST with different number of neighbors, denoted by N_s , involved. From Tab.2, we see that once the graph information is involved, the performance of the proposed method is significantly improved immediately, even only a few of neighbors are involved. Meanwhile, with increase of the number of the neighbors, the clustering accuracy keep improved, until too many neighbors are included.

5. Conclusion

We have proposed a graph embedding variational GMM for clustering. We have proposed stochastic graph embedding to impose a regularization on the pair of samples that connected on graph to push them to have similar posterior distributions. The similarity was measured by the Jensen-Shannon (JS) divergence, and an upper bound was derived to enable efficient learning. The proposed method outperforms deep model-based clustering and deep spectral clustering. Future work investigates extension with GAN discriminators [20, 34, 35].

Acknowledgement

This research is supported by the National Research Foundation Singapore under its AI Singapore Programme (Award Number: AISG-100E-2018-005). This work is also supported by both ST Electronics and the National Research Foundation (NRF), Prime Minister’s Office, Singapore under Corporate Laboratory at University Scheme (Programme Title: STEE Infosec - SUTD Corporate Laboratory). This work is supported in part by the National Science Foundation of China under Grant 61871091. Linxiao Yang is supported by the China Scholarship Council.

References

- [1] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. *Automatic subspace clustering of high dimensional data for data mining applications*, volume 27. ACM, 1998.

- [2] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.
- [3] Dimitri P Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.
- [4] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.
- [5] Deng Cai and Xinlei Chen. Large scale spectral clustering via landmark-based sparse representation. *IEEE transactions on cybernetics*, 45(8):1669–1680, 2015.
- [6] Dongdong Chen, Jiancheng Lv, and Yi Zhang. Unsupervised multi-manifold clustering by learning deep representation. In *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.
- [7] Nat Dilokthanakul, Pedro AM Mediano, Marta Garnelo, Matthew CH Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv preprint arXiv:1611.02648*, 2016.
- [8] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.
- [9] Dominik Maria Endres and Johannes E Schindelin. A new metric for probability distributions. *IEEE Transactions on Information theory*, 2003.
- [10] Charless Fowlkes, Serge Belongie, Fan Chung, and Jitendra Malik. Spectral grouping using the nystrom method. *IEEE transactions on pattern analysis and machine intelligence*, 26(2):214–225, 2004.
- [11] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.
- [12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [13] Xiaofei He and Partha Niyogi. Locality preserving projections. In *Advances in neural information processing systems*, pages 153–160, 2004.
- [14] Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama. Learning discrete representations via information maximizing self-augmented training. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1558–1567. JMLR. org, 2017.
- [15] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.
- [16] Bo Jiang, Chris Ding, Bio Luo, and Jin Tang. Graph-laplacian pca: Closed-form solution and robustness. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3492–3498, 2013.
- [17] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 1965–1972, 2016.
- [18] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [19] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(1):1, 2009.
- [20] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, ICML'16*, 2016.
- [21] Marc T Law, Raquel Urtasun, and Richard S Zemel. Deep spectral clustering learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1985–1994. JMLR. org, 2017.
- [22] Mu Li, James T Kwok, and Bao-Liang Lu. Making large-scale nyström approximation possible. In *ICML*, pages 631–638, 2010.
- [23] Xiaopeng Li, Zhourong Chen, Leonard KM Poon, and Nevin L Zhang. Learning latent superstructures in variational autoencoders for deep multidimensional clustering. In *7th International Conference on Learning Representations*, 2019.
- [24] Erxue Min, Xifeng Guo, Qiang Liu, Gen Zhang, Jianjing Cui, and Jun Long. A survey of clustering with deep learning: From the perspective of network architecture. *IEEE Access*, 6:39501–39514, 2018.
- [25] Maria CV Nascimento and Andre CPLF De Carvalho. Spectral methods for graph clustering—a survey. *European Journal of Operational Research*, 211(2):221–231, 2011.
- [26] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856, 2002.
- [27] Feiping Nie, Zinan Zeng, Ivor W Tsang, Dong Xu, and Changshui Zhang. Spectral embedded clustering: A framework for in-sample and out-of-sample spectral clustering. *IEEE Transactions on Neural Networks*, 22(11):1796–1808, 2011.
- [28] Feiping Nie, Wei Zhu, and Xuelong Li. Unsupervised large graph embedding. In *Thirty-first AAAI conference on artificial intelligence*, 2017.
- [29] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.
- [30] Edgar Schönfeld, Sayna Ebrahimi, Samarth Sinha, Trevor Darrell, and Zeynep Akata. Generalized zero-and few-shot learning via aligned variational autoencoders. *arXiv preprint arXiv:1812.01784*, 2018.
- [31] Uri Shaham, Kelly Stanton, Henry Li, Boaz Nadler, Ronen Basri, and Yuval Kluger. Spectralnet: Spectral clustering using deep neural networks. In *Sixth International Conference on Learning Representations*, 2018.
- [32] Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. Style transfer from non-parallel text by cross-alignment. In *Advances in neural information processing systems*, pages 6830–6841, 2017.

- [33] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888 – 905, 2000.
- [34] Tran Ngoc Trung, Tuan Anh Bui, and Ngai-Man Cheung. Dist-gan: An improved gan using distance constraints. In *ECCV*, 2018.
- [35] Tran Ngoc Trung, Tuan Anh Bui, and Ngai-Man Cheung. Improving gan with neighbors embedding and gradient matching. In *AAAI*, 2019.
- [36] René Vidal. Subspace clustering. *IEEE Signal Processing Magazine*, 28(2):52–68, 2011.
- [37] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487, 2016.
- [38] Rui Xu and Donald C Wunsch. Survey of clustering algorithms. 2005.
- [39] Donghui Yan, Ling Huang, and Michael I Jordan. Fast approximate spectral clustering. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 907–916. ACM, 2009.
- [40] Shuicheng Yan, Dong Xu, Benyu Zhang, Hong-Jiang Zhang, Qiang Yang, and Stephen Lin. Graph embedding and extensions: A general framework for dimensionality reduction. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (1):40–51, 2007.
- [41] Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3861–3870. JMLR. org, 2017.
- [42] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5147–5156, 2016.