2014 22nd International Conference on Pattern Recognition

Deep Embedding Network for Clustering

Peihao Huang, Yan Huang, Wei Wang, Liang Wang
Center for Research on Intelligent Perception and Computing (CRIPAC)
National Laboratory of Pattern Recognition, Institute of Automation
Chinese Academy of Sciences, Beijing 100190, China
{*phhuang, yhuang, wangwei, wangliang*}*@nlpr.ia.ac.cn*

***Abstract*—Clustering is a fundamental technique widely used for exploring the inherent data structure in pattern recognition and machine learning. Most of the existing methods focus on modeling the similarity/dissimilarity relationship among instances, such as k-means and spectral clustering, and ignore to extract more effective representation for clustering. In this paper, we propose a deep embedding network for representation learning, which is more beneficial for clustering by considering two constraints on learned representations. We first utilize a deep autoencoder to learn the reduced representations from the raw data. To make the learned representations suitable for clustering, we first impose a locality-persevering constraint on the learned representations, which aims to embed original data into its underlying manifold space. Then, different from spectral clustering which extracts representations from the block diagonal similarity matrix, we apply a group sparsity constraint for the learned representations, and aim to learn block diagonal representations in which the nonzero groups correspond to its cluster. After obtaining the learned representations, we use k-means to cluster them. To evaluate the proposed deep embedding network, we compare its performance with k-means and spectral clustering on three commonly-used datasets. The experiments demonstrate that the proposed method achieves promising performance.**

I. Introduction

It is very important to learn good features for achieving good performance in computer vision and pattern recognition tasks. Taking object recognition for an example, inspired by human visual system, an object can be represented by multiple level features, e.g., bar-like edges or object parts. Generally, the high-level features are made up of the low-level ones, which can achieve better recognition performance since they are more closely related to the abstract semantic concept. To obtain these multiple levels of features, many efforts have been put forward to design various hierarchical feature extractors. Deep neural network (DNN), as a typical hierarchical model, had been utilized for feature learning for a period of time in the 90's, which was abandoned afterwards due to the disadvantages of high computational cost, easily getting stuck in local optimum and over-fitting.

However, DNN has recently attracted much attention again since Hinton et al. [1] proposed an efficient learning algorithm for so-called deep belief nets (DBN). The algorithm utilizes a layer-wise unsupervised learning strategy to provide a good initialization for the network, which can greatly alleviate the drawbacks mentioned above. Thus, variants of DNN have exhibited their powerful ability in representation learning and achieved great success in many fields. For example, Lee et al. [2] propose a convolutional version of DBN to learn hierarchical representations for high-dimensional images, which obtain great improvement in visual recognition tasks. To fuse multiple modalities and obtain their shared representations, Ngiam et al. [3] propose a multimodal deep auto-encoder which can be learned by pretraining and fine-tuning successively. Krizhevsky et la. [4] design a very deep convolutional network for large-scale high-resolution image classification which largely outperforms the previous state-of-the-art. To better model the emission distribution of hidden Markov models, Mohamed et al. [5] replace the Gaussian mixture models with DBN and achieve better speech recognition results. However, to the best of our knowledge, there exists few literature which applies the widely used deep neural network for clustering.

Clustering can be considered as one of the most important unsupervised learning problems, which has been widely used in various fields from computer science to social science. The goal of clustering is to group similar data into the same cluster through measuring the distances between data points. Traditionally, k-means[6] is one of the most popular and simplest clustering methods, which iteratively assigns data to its nearest centroid and updates k centroids until convergence. Unlike the hard assigning of k-means, the mixture of Gaussians models each cluster as Gaussian distribution and represents the entire data set by a mixture of Gaussians. Without making any assumption on the distribution of the clusters, spectral clustering [7] makes use of the spectrum of the similarity matrix of data to perform dimensionality reduction before applying k-means. Most of the existing clustering methods focus on modeling the similarity/dissimilarity relationship among instances and ignore to extract more effective representation which largely influences the clustering performance. A few kernel methods, such as kernel k-means [8], apply a non-linear transformation to original data and generally measure the similarity in a high-dimensional representation space. However, these kernel methods heavily depend on the choice of the kernel, which is more experience-guided.

In this paper, we propose a deep neural network based model named deep embedding network (DEN), which learns clustering-oriented representations by imposing two special constraints. First, utilizing the deep autoencoder [9] to obtain good representations from the raw data by minimizing the data reconstruction error. To uncover the intrinsic manifold from the learned representations, we apply a locality-preserving constraint which preserves the local structure property of the original data. To further facilitate the clustering and make the representations contain cluster information, we also use a group sparsity constraint which aims to diagonalize the affinity of representations. The nonzero values of the representations correspond to its cluster. After obtaining the

1051-4651/14 $31.00 © 2014 IEEE
DOI 10.1109/ICPR.2014.272

learned representations, we use k-means to cluster them. To evaluate the proposed deep embedding network, we compare its performance with k-means and spectral clustering on three commonly-used datasets. The experiments demonstrate that the proposed method achieves promising performance.

The rest of this paper is organized as follows: Section II will introduce some related work about deep learning and clustering methods. The proposed deep embedding network for clustering will be described in detail in Section III followed by some experimental results in Section IV. We conclude the paper in Section V.

II. Related Work

In this section, we briefly review the existing literature that closely relates to the proposed deep embedding network. First, we will introduce the existing deep neural network methods for clustering. Then, we review current deep neural network methods for nonlinear embedding, which aims to learn robust representations for various tasks. Considering that binary units are used in the top hidden layer of the deep embedding network, deep embedding network seems to become able to learn hash codes, and some related networks need to be introduced below.

Deep neural network for clustering Currently, there is only a little work on deep neural network for clustering. Song et al. [10] propose an autoencoder-based data clustering method, which defines a new objective function by considering the reconstruction error from an auto-encoder network and restricting the distance in the learned space between data and their corresponding cluster centers. This model follows the two-step iteration procedure of k-means. During optimization, data representation and clustering centers are updated iteratively. Different with their work, our deep embedding network aims to learn effective representation for clustering, but not to simulate any existing clustering methods.

Deep neural network for nonlinear embedding Traditional autoencoder is an artificial neural network method for dimensionality reduction, which aims to learn a compressed representation for an input through minimizing its reconstruction error [11]. More recently, several autoencoder variants are proposed to learn robust features. Sparse autoencoder imposes a sparsity constraint on hidden representation to model the sparse coding of primary visual cortex [12]. Denoising autoencoder transforms a corrupted input into a hidden representation, and then tries to recover the original input from this representation [13]. Contractive autoencoder adds a regularization term that is the sum of squares of all partial derivatives of the hidden representations with respect to the input, which penalizes the sensitivity of the hidden representation to the input [14]. Hinton et al. propose an efficient pretraining algorithm for multilayer autoencoders (so-called deep autoencoder) [15]. In our deep embedding network, we use multilayer autoencoder to model the reconstruction term, and also adopt the pretraining strategy in [9]. Another similar work to our deep embedding network is the nonlinear extension of neighborhood component analysis (nonNCA) [16][17], which uses a deep neural network as the non-linear transformation function and preserves the class neighborhood structure by exploiting class label information. The main difference between our method and nonNCA is that our deep embedding network is an unsupervised representation learning method while nonNCA is a supervised one.

Deep neural network for learning hash codes Both [18] and [19] use deep autoencoder to learn hash codes by assuming that the top hidden layer only contains binary units. The former, [18] aims to learn a representation for each document while the later [19] for images. During the fine-tuning procedure, they use backpropagation to find codes that are good at reconstructing the word-count vector or original raw-pixel image but are as close to binary as possible. In order to make the codes binary, they add Gaussian noise to the bottom-up input received by each code unit, and make these bottom-up inputs to be large and negative for some training cases while large and positive for the others. With this binarization strategy, our deep embedding network can also be used to learn hash codes.

III. Clustering with Deep Embedding Network

Existing popular clustering methods such as k-means or spectral clustering, use either raw or linear transformed data for clustering. However, it would be insufficient when dealing with most datasets which have complex statistical properties. Recently, deep learning has drawn much attention because its highly nonlinear architecture can help to learn powerful feature representations. Thus, to take advantage of it, we propose a deep embedding network which utilizes deep neural network to learn clustering-oriented representations from raw data, and then performs clustering with k-means. To achieve the clustering-oriented representations, we impose two constraints on the learned representations of deep embedding network. One is a locality-persevering constraint which aims to embed original data into its underlying manifold space. The other one is a group sparsity constraint which aims to learn block diagonal representations in which the nonzero groups correspond to its cluster.

A. Deep Autoencoder

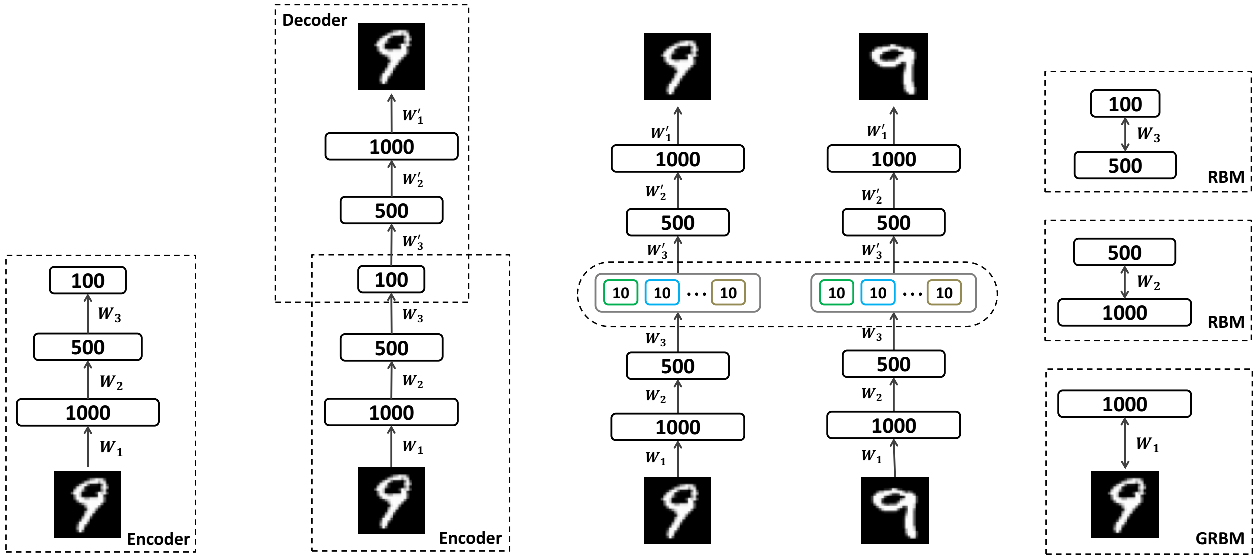
In this section, we will briefly review the traditional deep autoencoder (DAE) [9], which is the foundation of our proposed deep embedding network.

Taking the four-layer neural network in Fig. 1 (a) for example, the network serves as an encoder which transforms raw input data (digit 9) to a powerful representation (top layer with 100 units). Particularly, let $\mathbf{X} = \{\mathbf{x}_i : \mathbf{x}_i \in \mathbb{R}^{n\times 1}\}_{i=1,2,\cdots,N}$ denote the set of input data, and $W = \{W_i\}_{i=1,2,3}$ for the weights of the encoder network. The encoder defines a transformation $f(\cdot) : \mathbb{R}^{n\times 1} \rightarrow \mathbb{R}^{d\times 1}$ which transforms an input $\mathbf{x}$ to a d-dimensional representation $f(\mathbf{x})$:

$$f(\mathbf{x}) = W_3^T \phi(W_2^T \phi(W_1^T \mathbf{x})) \tag{1}$$

where $\phi(\cdot)$ is the activation function $\phi(\mathbf{x}) = 1/(1 + e^{-\mathbf{x}})$. For simplicity, we drop the bias term b_i for each layer in the formulation.

Unrolling the four-layer encoder network, we obtain a deep autoencoder with a symmetric decoder as shown in Fig. 1 (b). The decoder also defines a transformation $\hat{f}(\cdot) : \mathbb{R}^{d\times 1} \rightarrow \mathbb{R}^{n\times 1}$ which uses the transformed representation $f(\mathbf{x})$



(a) Deep neural network (b) Unrolling the network (c) The proposed deep embedded network (d) Pretraining

Fig. 1. (a) A four-layer deep neural network. The number in each layer represents the number of units, W_1, W_2 and W_3 are the weights of the network, the bottom digit image represents the input data, and the top layer with 100 units is the learned representation. (b) Unrolling the network. The encoder and decoder are marked as two dashed rectangles, the top digit image is the reconstruction of the bottom digit image. (c) The proposed deep embedding network. The top layer with 100 units is divided into 10 groups, each of which contains 10 units. (d) The pretraining procedure. The network can be trained successively as a GRBM and two RBMs.

to reconstruct the original input $\mathbf{x}$. When the input is binary-valued, the reconstructed input is

$$\hat{f}(\mathbf{x}) = \phi(W_1^T\phi(W_2^T\phi(W_3^T f(\mathbf{x})))) \tag{2}$$

When the input is real-valued, the reconstructed input is

$$\hat{f}(\mathbf{x}) = W_1^T\phi(W_2^T\phi(W_3^T f(\mathbf{x}))) \tag{3}$$

After obtaining the reconstructed input, we can define an objective function E_r when the input is real-valued:

$$E_r = \sum_{i=1}^{N} \|\mathbf{x}_i - \hat{f}(\mathbf{x}_i)\|^2 \tag{4}$$

where $\|\cdot\|$ denotes the Euclidean norm. When given binary-valued input, the corresponding objective function is defined as cross entropy:

$$E_r = -\sum_{i=1}^{N}[\mathbf{x}_i \log \hat{f}(\mathbf{x}_i) + (1-\mathbf{x}_i)\log(1-\hat{f}(\mathbf{x}_i))] \tag{5}$$

The network weights $W = \{W_i\}_{i=1,2,3}$ can be learned via gradient descend. We can compute the derivatives of the objective function with respect to all the weights using the backpropagation algorithm [9].

B. Objective Function of Deep Embedded Network

Deep neural network and its variants have been successfully applied to many pattern recognition tasks such as dimensionality reduction, classification and retrieval([14], [20], [18]). They can learn expressive representations for very complex data with their deep and nonlinear architectures. In this paper, we aim to learn clustering-oriented representations by imposing locality-preserving and group sparsity constraints.

Locality-preserving Constraint: To make the learned representations suitable for clustering, they are expected to be embedded into their intrinsic manifold, which preserves their local property. Specifically, we introduce a locality-preserving constraint as follows:

$$E_g = \sum_{i,j\in k(i)} S_{ij}\|f(\mathbf{x}_i) - f(\mathbf{x}_j)\|^2 \tag{6}$$

where $k(i)$ is the set containing the indexes of k nearest neighbors of data $\mathbf{x}_i$, $f(\cdot)$ is defined in (1), and S_{ij} is the similarity measure between $\mathbf{x}_i$ and $\mathbf{x}_j$. Here we use the heat kernel $S_{ij} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{t}}$ (t is a tuning parameter).

The manifold learned from the above constraint is similar to the one learned from Laplacian Eigenmap (LE) due to the similar formulation. As we know, LE is closely related to spectral clustering[21]. That is to say, our method is related to spectral clustering to some extent, except the nonlinear transformation implemented by deep neural network.

Group Sparsity Constraint: Inspired by spectral clustering which exploits block diagonal similarity matrix for representation learning, we aim to diagonalize the learned representations to further facilitate clustering by selecting one or more relevant groups of hidden units which represents inherent properties of data. As we know, group sparsity constraint as an effective feature selection method which has been widely used in many applications. In this paper, we employ group lasso which leads to sparsity within hidden codes on group level[22].

In particular, we divide the hidden units into G groups where G is the assumed number of clusters. When given a data point $\mathbf{x}_i$, we obtain the transformed representation $f(\mathbf{x}_i)$ and G grouped units $\{f^g(\mathbf{x}_i)\}_{g=1}^G$. Thus the group sparsity constraint E_s can be defined as follows[1]:

$$E_s = \sum_{i=1}^{N}\sum_{g=1}^{G}\lambda_g \| f^g(\mathbf{x}_i)\| \tag{7}$$

where $\{\lambda_g\}_{g=1}^G$ are the weights to sparsity of groups. Generally, larger weights are given to larger groups and λ_g is defined as follows:

$$\lambda_g = \lambda\sqrt{n_g}$$

where n_g is the group size and λ is a constant.

As we can see, by imposing this group sparsity constraint, only a few groups of the learned representation of each data point can be activated, and the activated groups correspond to specific cluster. As a result, all the representations can be block-diagonalized.

Objective function: After introducing the locality-preserving and group sparsity constraints in detail, we can derive the objective function of the proposed deep embedding network as follows:

$$E = E_r + \alpha E_g + \beta E_s \tag{8}$$

where α and β are tuning parameters.

C. Learning

To learn the network weights of the proposed deep embedding network, we utilize a two-stage algorithm which contains a pretraining procedure to initialize the network weights followed by a fine-tuning procedure. It should be noticed that both of these two procedures are unsupervised.

At first, we briefly review restricted Boltzmann machine (RBM) [9], which is a basic concept in pretraining. A RBM consists of a visible layer and a hidden layer. Each node in the visible layer is connected to each node in the hidden layer, and values of these nodes are all binary-valued. The energy function of this model is defined as follows:

$$F(\mathbf{v},\mathbf{h}) = -\mathbf{v}^T W\mathbf{h} - \mathbf{b}_1\mathbf{v} - \mathbf{b}_2\mathbf{h} \tag{9}$$

where $\mathbf{v}$ and $\mathbf{h}$ are respectively the visible and hidden nodes, W is the weight matrix between visible nodes and hidden nodes, $\mathbf{b}_1$ and $\mathbf{b}_2$ are the visible biases and hidden biases, respectively.

RBM can only deal with binary-valued data, while Gaussian restricted Boltzmann machine (GRBM) can handle real-valued data using real-valued visible nodes. Its energy function is defined as:

$$F(\mathbf{v},\mathbf{h}) = \sum_i \frac{(v_i-b_i)^2}{2\sigma_i^2} - \sum_i\sum_j \frac{v_i}{\sigma_i}W_{ij}h_j - \sum_j b_j h_j \tag{10}$$

Where $\{W, b_i, b_j\}$ are model parameters, σ_i is the standard deviation of the Gaussian noise for visible node i. The joint probability distribution of all the nodes is defined as:

$$P(\mathbf{v},\mathbf{h}) = \frac{1}{Z}\exp(-F(\mathbf{v},\mathbf{h})) \tag{11}$$

where Z is a normalization factor that scales $P(\mathbf{v},\mathbf{h})$ to [0,1]. The RBM parameters can be trained by minimizing the negative log-likelihood $-\sum_{\mathbf{h}}\log P(\mathbf{v},\mathbf{h})$ via stochastic gradient descend. Here we use Contrastive Divergence (CD) [15] to approximate the intractable gradients computation.

Pretraining [9] treats adjacent layers as a RBM and trains RBMs bottom-up to obtain good initial weights. For example, in Fig. 1 (d), the weight W_1 can be trained by treating the bottom two layer as a GRBM. The weight W_2 can be trained in the same way by treating the following two layers as a RBM. Similarly, we can train all the weights in a bottom-up manner. After the pretraining procedure, we use the backpropagation algorithm to fine-tune the network weights.

D. K-means for Clustering

Given data points, we first utilize the trained deep neural network to obtain the transformed representations, and then employ traditional k-means algorithm to perform clustering.

IV. Experimental Results

To evaluate the proposed deep embedding network, we perform several experiments on three datasets to demonstrate the effectiveness of our method in this section.

A. Description of Datasets

We use three datasets (COIL-20, PIE, Yale-B) to evaluate the performance of the proposed deep embedding network. All of these datasets provide ground truth, such as digit label, object class and person id.

COIL-20[2] is a dataset for object classification on 20 classes of objects. Each object has 72 images shot in different angles. We use the processed version of this dataset in which every image only contains one object with black background. All images are rescaled into a size of 32×32 pixels.

CMU PIE [24] and Yale-B [25] are both face image datasets, which contains faces of different people under varied poses, illumination conditions and expressions (Yale-B does not consider expression variation). For computational consideration we resize the images in Yale-B[3] to 30×40 pixels. Images in PIE are preprocessed as in [26].

For all datasets listed above, we normalize the data by subtracting the mean and then dividing the data by standard deviation for each feature.

[1] Noting that for smoothness of the differential of the group lasso regularizer, it is common to add a small positive ϵ term to the regularizer in practice [23] . The formula of regularization term E_s then turns to $\sum_{g=1}^{G}\lambda_g\sqrt{\sum_{i\in\text{group } g} h_i^2+\epsilon}$.

[2] http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php

[3] http://markus-breitenbach.com/machine_learning_data.php

TABLE I. Comparison of NMI and Accuracy of clustering methods on three datasets

Method	COIL20		Yale-B		PIE	
	NMI	ACC (%)	NMI	ACC (%)	NMI	ACC (%)
K-means	0.76±0.02	59.02±4.40	0.70±0.04	65.80±6.18	0.19±0.00	8.36±0.40
Ncut	0.86±0.03	64.57±6.87	0.79±0.03	62.34±6.37	0.25±0.01	9.55±0.65
Proposed	**0.87±0.01**	**72.40±3.39**	**0.92±0.04**	**81.73±9.64**	**0.42±0.00**	**11.19±0.29**

B. Experimental Settings

1) Baselines: We compare the proposed method with two most popular clustering algorithms: k-means [6] and spectral clustering [7]. Considering that spectral clustering is an approximated solution for normalized cut [27] or ratio cut [28]. Here we take the normalized cut (Ncut) version of spectral clustering as the baseline.

2) Evaluation metric: As the group label of each data point is available, we can compare clustering results with these labels to evaluate the performance. Here we use Normalized Mutual Information (NMI) and Accuracy (ACC) to measure the effectiveness of clustering methods.

Normalized Mutual Information[29] is a popular metric used for evaluating clustering tasks. It is defined as follows:

$$\text{NMI}(\Omega, \mathbb{C}) = \frac{I(\Omega, \mathbb{C})}{\sqrt{H(\Omega)H(\mathbb{C})}}$$

where Ω and $\mathbb{C}$ respectively denote clustering label and ground truth label of any given sample. $I(\Omega, \mathbb{C})$ is mutual information which measures the information gain to the true partition after knowing the clustering result, $H(\cdot)$ is entropy and the denominator $\sqrt{H(\Omega)H(\mathbb{C})}$ is used to normalize the mutual information to be in the range of $[0, 1]$. When we partition the data perfectly, NMI score is 1, and when Ω and $\mathbb{C}$ are independent, NMI score is 0.

Accuracy is defined as:

$$\text{ACC} = \frac{\sum_{i=1}^{N} \delta(map(c_i) = y_i)}{N}$$

where $\delta(\cdot)$ is an indicator function, c_i is the clustering label for $\mathbf{x}_i$, $map(\cdot)$ transforms the clustering label c_i to its group label by the Hungarian algorithm [30], and y_i is the true group label of $\mathbf{x}_i$. ACC measures the consistency between the true group label and the clustering group label.

Initial centroids have significant impact on clustering results when utilizing the k-means algorithm. To alleviate this influence, we repeat k-means for multiple times with random initial centroids (specifically, 100 times for statistical significance). The average NMI and accuracy along with the corresponding standard deviation are taken as the final result.

TABLE II. Parameter Settings on three datasets

Dataset	Architecture	K	s	α	β
COIL20	1024-200-80	20	4	250	0
Yale-B	1200-500-100-30	10	3	0.5	1
PIE	1024-2000-680	68	10	0	0

3) Parameter settings: We manually choose the architecture for each deep embedding network, such as the number of layers and the number of units in each layer. As for the groups of the top hidden layer codes in the deep embedding network, we group adjacent hidden codes together and fix the size of each group to a constant s (e.g., 3 or 5). Note that there is no intersection between groups. When partitioning the dataset into K clusters, we obtain $K \cdot s$ hidden codes in the network. Detailed settings for all datasets are listed in Table II.

C. Results Analysis

We train the deep embedding network on the datasets and apply the k-means algorithm on the learned representation. NMI and Accuracy of all the methods are listed in Table I.

The experimental results show that the proposed deep embedding network performs significantly better than both k-means and spectral clustering algorithms on all the tested datasets. It is because k-means can discover more accurate cluster structure from the representations learned by deep embedding network than that from raw data. Compared with normalized cut, our method achieves large improvement on all these datasets by 0.01/7.83%, 0.13/19.39%, and 0.17/1.64% (NMI/ACC), respectively. Note that for each dataset, we tune the parameters of normalized cut to obtain the best result (including the number of the nearest neighbors for graph construction and the edge weights).

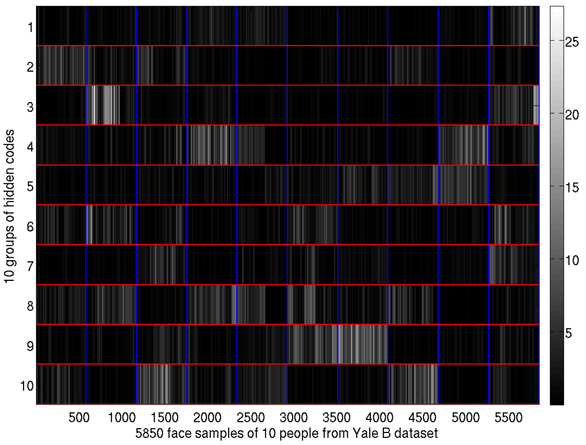


Fig. 2. Representations learned from 5,850 samples of Yale-B. Horizontal axis represents the samples while vertical axis represents groups in the learned representations. Blue lines separate groups in samples and red ones separate groups in the learned representations. (Best viewed in color.)

We also plot the learned representations of the 5,850 samples from the Yale-B dataset in Figure 2. In this figure, the horizontal axis denotes the samples and the blue lines separate the ten group samples from each other. The vertical axis denotes the learned representation and the red lines separate the ten group representations. As we can see, the representations in each sample group show the group sparsity, and different sample groups show different group sparsity patterns, which indicates that the proposed method learns good representations for clustering.

V. Conclusion

In this paper, we have proposed the deep embedding network for clustering, which exploits deep neural network to obtain clustering-oriented representations by considering two constraints, namely the locality-preserving and group sparsity constraints, respectively. The experimental results have demonstrated the effectiveness of our proposed method.

Acknowledgment

This work is jointly supported by National Natural Science Foundation of China (61175003, 61135002, 61202328), Hundred Talents Program of CAS, National Basic Research Program of China (2012CB316300).

References

[1] G. E. Hinton and S. Osindero, "A fast learning algorithm for deep belief nets," *Neural Computation*, 2006.

[2] H. Lee, R. Grosse, R. Ranganath, and A. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," *International Conference on Machine Learning*, 2009.

[3] J. Ngiam, A. Khosia, J. Nam, H. Lee, and A. Ng, "Multimodal deep learning," *International Conference on Machine Learning*, 2011.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, 2012.

[5] A. Mohamed, G. E. Dahl, and G. E. Hinton, "Acoustic modeling using deep belief networks," *IEEE TASLP*, 2012.

[6] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl, "Constrained k-means clustering with background knowledge," in *International Conference on Machine Learning*, vol. 1, 2001, pp. 577–584.

[7] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[8] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means, spectral clustering and normalized cuts," in *Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2004, pp. 551–556.

[9] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, 2006.

[10] C. Song, F. Liu, Y. Huang, and T. Tan, "Auto-encoder based data clustering," *Springer*, 2013.

[11] Y. Bengio, "Learning deep architectures for ai," *Foundations and Trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[12] H. Lee, C. Ekanadham, and A. Ng, "Sparse deep belief net model for visual area v2," *Advances in Neural Information Processing Systems*, 2007.

[13] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," *International Conference on Machine Learning*, 2008.

[14] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," *International Conference on Machine Learning*, 2011.

[15] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Computation*, 2002.

[16] J. Goldberger, S. T. Roweis, and G. E. Hinton, "Neighbourhood components analysis," *Advances in Neural Information Processing Systems*, 2004.

[17] R. Salakhutdinov and G. Hinton, "Learning a non-linear embedding by preserving class neighbourhood structure," *AI and Statistics*, 2007.

[18] R. Salakhutdinov and G. E. Hinton, "Semantic hashing," *IJAR*, 2009.

[19] A. Krizhevsky and G. E. Hinton, "Using very deep autoencoders for content-based image retrieval." in *ESANN*. Citeseer, 2011.

[20] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, 2012.

[21] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering." in *Advances in Neural Information Processing Systems*, vol. 14, 2001, pp. 585–591.

[22] L. Meier, S. Van De Geer, and P. Bühlmann, "The group lasso for logistic regression," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 70, no. 1, pp. 53–71, 2008.

[23] F. Nie, H. Huang, X. Cai, and C. Ding, "Efficient and robust feature selection via joint l_{21}-norms minimization," *Advances in Neural Information Processing Systems*, 2010.

[24] T. Sim, S. Baker, and M. Bsat, "The cmu pose, illumination, and expression (pie) database," in *Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on*. IEEE, 2002, pp. 46–51.

[25] A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman, "From few to many: Illumination cone models for face recognition under variable lighting and pose," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, no. 6, pp. 643–660, 2001.

[26] X. He, S. Yan, Y. Hu, P. Niyogi, and H.-J. Zhang, "Face recognition using laplacianfaces," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 3, pp. 328–340, 2005.

[27] J. Shi and J. Malik, "Normalized cuts and image segmentation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 8, pp. 888–905, 2000.

[28] S. Wang and J. M. Siskind, "Image segmentation with ratio cut," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, no. 6, pp. 675–690, 2003.

[29] W.-Y. Chen, Y. Song, H. Bai, C.-J. Lin, and E. Y. Chang, "Parallel spectral clustering in distributed systems," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 3, pp. 568–586, 2011.

[30] C. H. Papadimitriou and K. Steiglitz, *Combinatorial optimization: algorithms and complexity*. Courier Dover Publications, 1998.