ADCONIP 2014 Hiroshima

Latent Subspace Clustering based on Deep Neural Networks

Yanjun Ma, Chao Shang, Fan Yang, Dexian Huang*

***Abstract*— Clustering approaches have been widely used in process control community for unsupervised classification beneficial for further analysis, modeling and optimization. Process data generally involve far more dimensions than needed; this phenomenon is called as "data rich but information poor" and becomes obstacles for reasonable classification. Therefore, it is desirable to use latent variable models such as principal component analysis (PCA) to lower the dimension of data. Traditional clustering models, however, are directly established on the data and make no allowance for latent subspace, which would cause inaccuracy in unsupervised data classification. In recent years deep neural networks (DNN) have proved effective for developing latent variable models, which is termed as the "deep learning" technique. In this paper, we propose a novel clustering approach based on a combination of DNN and traditional K-means method. DNN is responsible for latent subspace description within the data, and the K-means method is used for clustering in the derived latent subspace. The proposed method has better generalization performance due to its strong nonlinear representation ability, and it is especially favored in the case of high-dimensional data with significant correlations. The efficacy of the proposed method is addressed on two benchmark data sets in comparison with traditional clustering approaches.**

I. INTRODUCTION

Multi-mode characteristics are commonly incurred in industrial processes mainly because they are operated under several different conditions [1]. Clustering techniques have established themselves as effective tools dealing with multi-mode phenomena. The main motivation of clustering is to classify data into different clusters according to certain similarity criteria. Data belonging to the same cluster are seen as yielding the same simpler local distribution. Therefore a task dealing with complicately distributed data can be desirably simplified to several sub-tasks because of their simple local structures. Over the last few decades, clustering methods have been intensively applied in a proliferation of process control tasks, including system identification [2], [3], soft sensor development [4] and process monitoring [5].

The structure determination and parameter estimation of clustering models have always been prevailing issues to researchers, and diversified clustering models have already been proposed. The *K-means* was the first algorithm for clustering purposes proposed over 50 years ago and it still gets extensively used at present [6]. The Euclidean metric and Mahalanobis distance metric are typically used in K-means for evaluating the distance between samples and cluster centers. Consequently, K-means can only find

[*] Y. Ma, C. Shang, F. Yang, D. Huang, Tsinghua National Laboratory for Information Science and Technology, Department of Automation, Tsinghua University, Beijing, 100084, CHINA; D. Huang, corresponding author, huangdx@mail.tsinghua.edu.cn

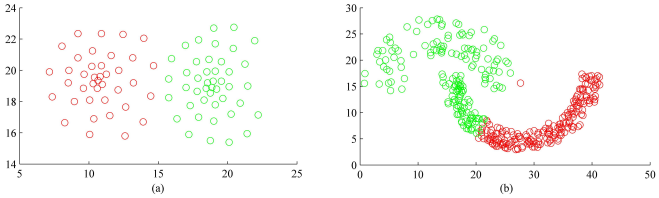

Fig. 1. Example of K-means performances in different scenarios: (a) spherical clusters; (b) non-spherical clusters.

spherical or hyper-ellipsoidal clusters, and fails to handle non-convex clusters, as shown in Fig. 1. In this regard, some extentions have been proposed such as *kernel K-means* in [7] and *support vector clustering* (SVC) in [8]. They use kernel tricks to project data onto a high-dimensional feature space and further cluster data in the feature space. The feature space, however, is often of extremely high or even infinite dimension, hence data distribution in the feature space cannot be explicitly described. This gives rise to difficulties in further analysis. Moreover, the kernel matrix computation becomes inconvenient and even intractable in the context of massive data.

Another clustering approach dealing with non-convex clusters is *self organizing map* (SOM) by [9]. It reduces data to a two-dimensional visualized structure by means of neural networks and further clusters data on the two-dimensional space. SOM has found wide applications in process visualization and fault isolation. Nevertheless, SOM lacks adequate representation capability dealing with high-dimensional data because all data are forced to spread on a two-dimensional space with some important information ignored.

In industrial processes, evident correlations are involved in high dimensional process data. The dimension of data is usually much higher than its effective dimension, which is termed as "data rich but information poor" [10]. This is mainly due to two reasons. First, the popularity of distributed control systems (DCSs) makes it possible to archive large amounts of data. Second, visible variations of data are caused by a smaller number of inherent process changes. Therefore, latent variable models, such as principal component analysis (PCA), are broadly employed to describe the latent subspace that explains most variances of data. It is latent variables that reflect the nature of data structure. Traditional clustering approaches, however, are not dependent on those intrinsic latent variables and thus fail to take underlying features into considerations.

Consequently, traditional approaches may lose efficacy

in the presence of high dimensional process data that are heavily correlated, and it is necessary to develop clustering models on the basis of low dimensional latent subspace of special interest. To this end, a novel clustering method based on latent subspace description has been proposed. In recent years, *deep neural networks* (DNN) have gained increasing attention in machine learning area [11], [12], [13], and have been successfully applied to dimensionality reduction and nonlinear feature subspace description because of their remarkable representation ability. In this study, such nonlinear subsupace clustering method mainly includes two steps. In the first step, the nonlinear subspace is extracted by using DNN. In the second step, the K-means-based clustering performed on the derived subspace is further integrated with DNN learning, in which the objective functions of both clustering and dimensionality reduction are combined and optimized together. In this way, the clustering model not only classifies data into different categories but also maintains the latent subspace containing most information in data.

The rest of this paper proceeds as follows. In the next section a brief review of K-means and the DNN based dimensionality reduction approach is presented. In Section III, the proposed latent subspace clustering model is given and its learning algorithm is detailed. In Section IV, the proposed method is tested on two benchmark datasets in machine learning community in comparison with traditional clustering approaches, followed by a brief conclusion in Section V.

II. PRELIMINARIES

A. K-means clustering revisit

The K-means clustering assumes in a priori that there are K clusters $C = \{C_1, C_2, \cdots, C_K\}$. Given a set of n-dimensional samples $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}$, K-means aims at minimizing the within-cluster sum of squares:

$$\arg\min_{C} \sum_{k=1}^{K} \sum_{\boldsymbol{x}_m \in C_k} ||\boldsymbol{x}_m - \boldsymbol{\mu}_k||^2, \tag{1}$$

where $\boldsymbol{\mu}_k$ is the mean of cluster C_k. Because minimizing (1) is known as an NP-hard problem, K-means can only converge to a local optimum. One effective strategy to avoid local optimum is to run K-means several times with random initialization and choose the one with least within-cluster sum of squares. The K-means algorithm includes the following iterative steps:

Step 1. Initialize K cluster centers $\{C_1, C_2, \cdots, C_K\}$ randomly.

Step 2. Assign each sample to its nearest cluster center and update the new partition.

Step 3. Derive new cluster centers $\{\boldsymbol{\mu}_k\}(k = 1, 2, \cdots, K)$ according to the current partition.

Step 4. If cluster partition changes, go back to Step 2. Otherwise, stop the algorithm.

B. Nonlinear subspace description using deep neural networks

1) Auto-encoder: An auto-encoder is a two-layer neural network with a hidden layer and an output layer [12]. Let $\boldsymbol{x}$ be an n-dimensional input vector with elements $x_i \in (0, 1)$ and $\boldsymbol{h}$ be an m-dimensional vector of the hidden layer. Fig. 2 gives a sketch of an auto-encoder. The hidden layer $\boldsymbol{h}$ is calculated as

$$\boldsymbol{h} = \mathrm{sigm}(W^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{b}), \tag{2}$$

where $\mathrm{sigm}(\boldsymbol{u})$ is the element-wise sigmoid activity function:

$$\mathrm{sigm}(\boldsymbol{u}) = \left[\frac{1}{1+\exp(-u_1)} \ \cdots \ \frac{1}{1+\exp(-u_m)}\right]^{\mathrm{T}}, \tag{3}$$

$W \in \mathbb{R}^{n\times m} and \boldsymbol{b} \in \mathbb{R}^{m\times 1}$ are a coefficient matrix and a bias vector, respectively. The vector $\hat{\boldsymbol{x}}$ of the output layer has dimension n, the same as the input vector. The output layer vector is calculated as

$$\hat{\boldsymbol{x}} = \mathrm{sigm}(W\boldsymbol{h} + \boldsymbol{c}), \tag{4}$$

where W is the same coefficient matrix as defined in (2) and $\boldsymbol{c}$ is a bias vector [12]. Notice that each element x_i of the input vector $\boldsymbol{x}$ is limited to the range $(0, 1)$. Therefore, such an auto-encoder is termed as a *binary* auto-encoder. To deal with input values in an unlimited range, a *Gaussian* auto-encoder is formulated as:

$$\begin{aligned} \boldsymbol{h} &= \mathrm{sigm}(W^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{b}), \\ \hat{\boldsymbol{x}} &= W\boldsymbol{h} + \boldsymbol{c}, \end{aligned} \tag{5}$$

where $x_i \in \mathbb{R}$. The auto-encoder aims at reconstructing its inputs in the output layer by minimizing the following reconstruction error function:

$$J(\theta) = ||\boldsymbol{x} - \hat{\boldsymbol{x}}||^2, \tag{6}$$

where θ represents the network parameters $\theta = \{W, \boldsymbol{b}, \boldsymbol{c}\}$.

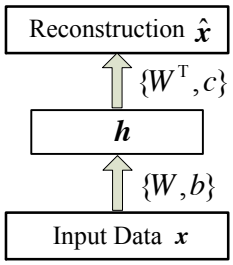

Fig. 2. Sketch of an auto-encoder

Traditional back-propagation can be used to train auto-encoders. In this study, the gradient descent approach is adopted with parameters randomly initialized. Notice that the auto-encoder is forced to learn an identity mapping in order to keep most information in the hidden layer.

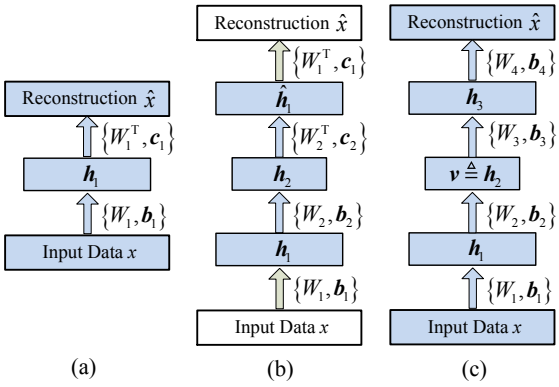

Fig. 3. Layer-wise training procedure of DNN: (a) initialize the auto-encoder θ_1; (b) train the auto-encoder θ_2 based on $\boldsymbol{h}_1$; (c) adjust the weights of DNN according to the reconstruction criterion.

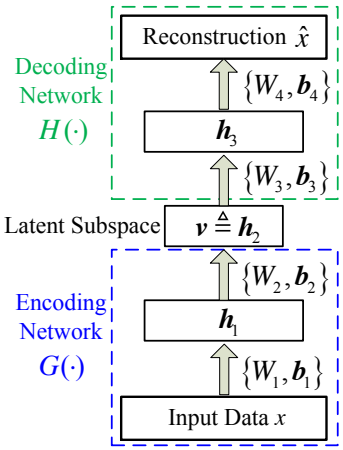

Fig. 4. Latent subspace derived from a five-layer DNN

2) Deep neural network based latent subspace description: Neural networks have already been utilized to deal with the dimensionality reduction problem two decades ago. Kramer proposed a five-layer neural network called the *autoassociative neural network*, which comprises an encoding network and a decoding network [14], [15]. Unfortunately, such autoassociative neural network is very difficult to train because of its complicated structure. The deep neural network (DNN), however, overcomes the difficulty successfully [11]. DNN is a generalized multi-layer model including one visible layer and several latent layers, which is made up of several auto-encoders that are connected in series. Fig. 3 gives a simple example of a five-layer DNN.

The learning of a DNN is a greedy process. As shown in Fig. 3, the auto-encoder θ_1 is trained first, and the second auto-encoder θ_2 is inserted to the latent layer $\boldsymbol{h}_1$ of θ_1. Auto-encoder θ_2 takes the latent layer $\boldsymbol{h}_1$ in θ_1 as its input vector and the goal is to reconstruct $\boldsymbol{h}_1$ in its output layer. Such a procedure can be repeated as many times as desired. The entire DNN is established by layer-wise training of each individual auto-encoder. It is highlighted that in DNN learning, high-level latent layers for high-level features are developed based on low-level latent layers for low-level features. It makes DNN a hierachical model with latent variables.

After training each individual auto-encoder, the DNN needs to be fine-tuned by reconstructing its input in the output layer. Notice that each auto-encoder realizes approximated input reconstruction; therefore parameters of DNN only need minor adjustments to make output resemble its input and a *conjugate gradient* (CG) optimization is adopted. In general, the hidden layer in the middle of DNN is set to have the lowest dimension, amounting to a "bottleneck" layer. Therefore the bottleneck layer with lowest dimension can be seen as a nonlinear subspace that explains most variance within data.

III. CLUSTERING BASED ON NONLINEAR LATENT SUBSPACE DESCRIPTION

A. Model and Objective

Assume that the DNN has $(2L+1)$ layers, and the coefficient matrix and the bias vector in each layer are defined as W_l and $\boldsymbol{b}_l$ $(l = 1, 2, \cdots, 2L)$, respectively. In practice, the input values of DNN are unlimited; thus the *Gaussian* auto-encoder is served as an input auto-encoder. The rest auto-encoders are naturally *binary* auto-encoders and back-propagation is performed when a new auto-encoder is added to the DNN. Each hidden layer can therefore be expressed as

$$\begin{aligned} &\boldsymbol{h}_0 = \boldsymbol{x}, \\ &\boldsymbol{h}_l = \mathrm{sigm}(W_l^{\mathrm{T}} \boldsymbol{h}_{l-1} + \boldsymbol{b}_l), l = 1, 2, \cdots, 2L-1, \\ &\boldsymbol{h}_{2L} = W_{2L}^{\mathrm{T}} \boldsymbol{h}_{2L-1} + \boldsymbol{b}_{2L}. \end{aligned} \tag{7}$$

The reconstructed output of DNN is defined as $\hat{\boldsymbol{x}} = \boldsymbol{h}_{2L}$, whereas the bottleneck layer of special interest is defined as $\boldsymbol{v} = \boldsymbol{h}_L$. For simplicity, we denote the *encoding network* from input $\boldsymbol{x}$ to bottleneck layer $\boldsymbol{v}$ as $\boldsymbol{v} = G(\boldsymbol{x})$, which has parameters $\{W_l, \boldsymbol{b}_l\}(l = 1, \cdots, L)$, and the *decoding network* from bottleneck layer $\boldsymbol{v}$ to its reconstruction output $\hat{\boldsymbol{x}}$ is defined as $\hat{\boldsymbol{x}} = H(\boldsymbol{v})$, which has parameters $\{W_l, \boldsymbol{b}_l\}(l = L+1, \cdots, 2L)$, as shown in Fig. 4. Assume that there are N samples in the training set, denoted as $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}$. The learning objective of DNN, namely, the minimization of the reconstruction error, is represented as:

$$J_1 = \frac{1}{N} \sum_{i=1}^{N} \|\hat{\boldsymbol{x}}_i - \boldsymbol{x}_i\|^2 = \frac{1}{N} \sum_{i=1}^{N} \|H(G(\boldsymbol{x}_i)) - \boldsymbol{x}_i\|^2. \tag{8}$$

.

After a latent subspace is derived by training deep neural network, clustering is to be executed on the latent subspace $\boldsymbol{v}$. The idea of K-means is employed, and an alternative cluster measure to (1), namely, the Davies-Bouldin validity index [16], is used. Assume that $\boldsymbol{\mu}_k$ is the center of cluster

C_k, and T_k is the number of elements in C_k. The learning objective of clustering is then defined as:

$$J_2 = \frac{1}{K}\sum_{k=1}^{K}\max\left\{\frac{S_k + S_j}{M_{kj}} \mid j \neq k\right\}, \tag{9}$$

where S_k is defined as the *distance inside each clusters*:

$$\begin{aligned} S_k &= \sqrt{\frac{1}{T_k}\sum_{\boldsymbol{v}_m \in C_k}\|\boldsymbol{v}_m - \boldsymbol{\mu}_k\|^2} \\ &= \sqrt{\frac{1}{T_k}\sum_{G(\boldsymbol{x}_m)\in C_k}\|G(\boldsymbol{x}_m) - \boldsymbol{\mu}_k\|^2}, \end{aligned} \tag{10}$$

and M_{kj} is defined as the *distance between two clusters* C_k and C_j:

$$M_{kj} = \|\boldsymbol{\mu}_k - \boldsymbol{\mu}_j\|. \tag{11}$$

Such two metrics will prevent the bottleneck layer to be over-concentrated or over-dispersed, which is helpful for the stability of the training algorithm.

It can be seen from (9)-(11) that simply minimizing J_2 will adjust parameters of the encoding network $G(\boldsymbol{x})$, without updating parameters of the decoding network $H(\boldsymbol{v})$. For this reason, the whole DNN cannot guarantee a desirable reconstruction of its inputs, leading to the destruction of the latent subspace. As a consequnce, the latent subspace based clustering will become unreliable. It is reasonable that the latent subspace can be maintained during the clustering procedure. Here a combined clustering objective J is adopted:

$$J = J_1 + \lambda J_2, \tag{12}$$

where λ is a regularization parameter that balances the reconstruction objective and the clustering objective. On one hand, if λ is too small, the combined objective would take little consideration about clustering and simply reduces the data dimension. On the other hand, if λ is too large, the latent subspace cannot be properly preserved and thus clustering would be no good. Therefore it is necessary to properly choose the regularization parameter λ, and the cross-validation strategy can be used.

B. Details of parameters adjustments

It can be seen that the reconstruction criterion J_1 deals with both $G(\boldsymbol{x})$ and $H(\boldsymbol{v})$, while the clustering criterion J_2 is only relevant with the encoding network $G(\boldsymbol{x})$. To minimize the reconstruction criterion J_1, the adjustment of weights can be readily obtained via traditional back-propagation [17]. As for the clustering criterion, the partial derivative of J_2 is detailed as follows:

$$\frac{\partial J_2}{\partial \boldsymbol{v}_m} = (n_k + 1)\cdot\frac{\boldsymbol{\mu}_k - \boldsymbol{v}_m}{T_k \cdot S_k \cdot M_{k,j^*(k)}}, \tag{13}$$

where $\boldsymbol{v}_m = G(\boldsymbol{x}_m) \in C_k$, n_k is the number of times that cluster C_k serves as the nearest cluster to other clusters and

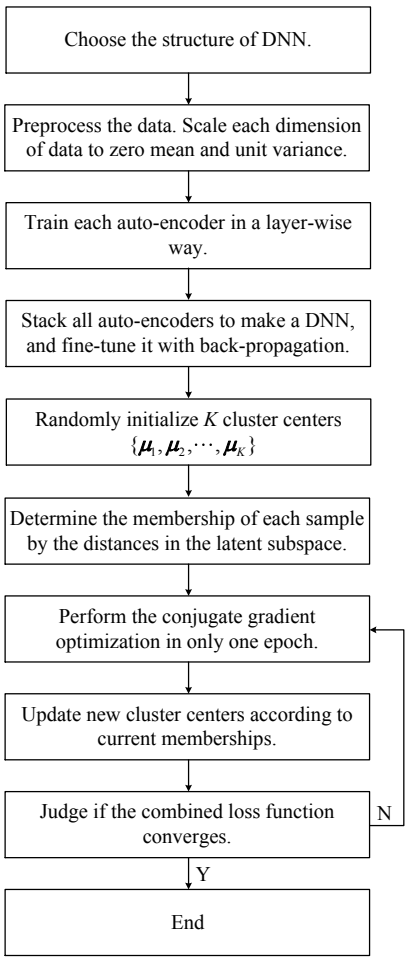

Fig. 5. Flowchart of DNN-based latent subspace clustering process.

$j^*(k) = \arg\max_j\{(S_k + S_j)/M_{kj} \mid j \neq k\}$. Then, the derivative to the parameters $\{W_l, \boldsymbol{b}_l\}(l = 1, \cdots, L)$ of the encoding network $G(\cdot)$ can be formulated as:

$$\begin{aligned} \frac{\partial J_2}{\partial W_l} &= \frac{1}{N}\sum_{i=1}^{N}\frac{\partial J_2}{\partial \boldsymbol{v}_i}\cdot\frac{\partial \boldsymbol{v}_i}{\partial W_l}, l = 1, \cdots, L, \\ \frac{\partial J_2}{\partial \boldsymbol{b}_l} &= \frac{1}{N}\sum_{i=1}^{N}\frac{\partial J_2}{\partial \boldsymbol{v}_i}\cdot\frac{\partial \boldsymbol{v}_i}{\partial \boldsymbol{b}_l}, l = 1, \cdots, L, \end{aligned} \tag{14}$$

where the terms $\partial \boldsymbol{v}_i/\partial W_l$ and $\partial \boldsymbol{v}_i/\partial \boldsymbol{b}_l$ can be obtained by the chain rule in the traditional back-propagation algorithm.

C. Latent subspace clustering based on DNN training

Because clustering is dependent on a well established latent subspace, the DNN should be built at the beginning. Afterwards, the combined objective in (8) can be pursued. If the network is at first optimized according to the combined objective J directly, it will get easily trapped into local optima because of structual complexity. The entire procedure of the algorithm is summarized in Fig. 5.

TABLE I

CLUSTERING ERROR RATES ON TWO BENCHMARK DATASETS (%)

	Our Method	K-means	SVC	SOM
IRIS dataset	2	10.67	9.33	2.67
WDBC dataset	4.04	8.96	9.31	5.80

IV. SIMULATION CASE STUDIES

In this section, two benchmark datasets, namely, the IRIS dataset and the Wisconsin diagnostic breast cancer (WDBC) dataset in [18], are used to testify the efficacy of the proposed method in comparison with traditional clustering methods.

The IRIS benchmark dataset describes iris plant features with four-dimensional samples. It contains 150 samples classified into three different categories. The WDBC dataset describes 30 features of the cell nuclei drawn from an image of a fine needle aspirate (FNA) of a breast mass. It includes 569 samples from two classes. Both datasets can be downloaded from http://archive.ics.uci.edu/ml/datasets/. All samples are classified into different clusters in an unsupervised manner. Because the class information is in a priori available, they can be used to calculate the numbers of falsely clustered samples and further give the error rate.

For IRIS dataset, a 4-15-8-3-8-15-4 DNN with seven layers is established, while a 30-100-50-20-50-100-30 DNN with five layers is developed for WDBC dataset. Because clustering in latent subsupace involves randomness, the learning process is repeated 100 times and the result with the best clustering performance is selected.

In this study, three traditional clustering approaches, namely, K-means, SVC, and SOM, are tested on two benchmark datasets for comparison purposes. Due to randomness in the training process of K-means and SOM, each of them is trained 100 times in the same way as our method and result with the best performance is selected. For SVC, the outlier parameter is 0.01 and the Gaussian kernel parameter is 1, according to cross-validation. For SOM, the topology size is 3×3. Table 1 gives the comparison results of our method as well as other three approaches. It can be seen that K-means gives the worst clustering results for both datasets, because it cannot deal with abnormal clusters. Our method and SOM, which are based on neural network techniques, clearly outperform the other two methods because neural networks enjoy powerful capability to represent complex clusters. In addition, our method achieves a smaller error rate than SOM, mainly because clustering is based on the derived latent subspace while SOM makes no allowance for latent information behind data.

Because the bottleneck layer of DNN for IRIS dataset is three-dimensional, it is convenient to visualize the latent subspace. Fig. 6(a) shows the latent subspace distribution after fine-tuning DNN but without clustering, and Fig. 6(b) shows the latent subspace distribution after clustering. On one hand, it is revealed in Fig. 6(a) that the latent subspace contains salient cluster information, even without using a clustering criterion. It demonstrates that DNN effectively reduces data dimension with most information maintained, thereby making clustering easier on a lower-dimensional subspace. On the other hand, by comparing Fig. 6(b) with (a), it can be seen that after clustering, samples in the same cluster gathers closely.

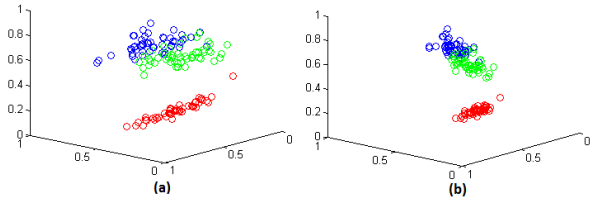

Fig. 6. Latent subspace distribution on IRIS dataset: (a) after DNN training but before clustering; (b) after clustering

In addition, we have attemped to skip the training phase of DNN and directly optimize the combined criterion for comparison purposes. It is found that in both IRIS and WDBC datasets, the networks are prone to local optima and have significantly poor generalization ability than our proposed strategy. This verifies the necessity of a pre-trained DNN in the latent subspace based clustering.

V. CONCLUSION

This paper proposed a clustering algorithm based on DNN. Once the DNN structure is defined, only one parameter remains in our algorithm, allowing it to obtain various solutions. A unique advantage of our algorithm is clustering of reduced-dimensional data that contains most of information in the original data. The novelty of our algorithm lies in that, the two stage training steps integrate the clustering index with reconstruction error, which make the unsupervised learning more intentional.

From experiments, we noticed the remaining nonlinearity in the bottleneck layer after DNN construction, which could lead to other nonlinear clustering methods to DNN. In addition, inspired by supervised learning of DNN, the clustering index may be helpful to determine the DNN's structure and learning process.

ACKNOWLEDGMENT

The authors would like to appreciate the funding from the National High-tech 863 Program of China (2013AA 040702), the National Natural Science Foundation of China (21276137), and Tsinghua University Initiative Scientific Research Program.

REFERENCES

[1] J. Kallrath, "Planning and scheduling in the process industry," *OR Spectrum*, vol. 24, no. 3, pp. 219–250, 2002.

[2] G. Ferrari-Trecate, M. Muselli, D. Liberati, and M. Morari, "A clustering technique for the identification of piecewise affine systems," *Automatica*, vol. 39, no. 2, pp. 205–217, 2003.

[3] H. Nakada, K. Takaba, and T. Katayama, "Identification of piecewise affine systems based on statistical clustering technique," *Automatica*, vol. 41, no. 5, pp. 905–913, 2005.

[4] M. Kim, Y.-H. Lee, I.-S. Han, and C. Han, "Clustering-based hybrid soft sensor for an industrial polypropylene process with grade changeover operation," *Industrial & Engineering Chemistry Research*, vol. 44, no. 2, pp. 334–342, 2005.

[5] P. Teppola, S.-P. Mujunen, and P. Minkkinen, "Adaptive fuzzy c-means clustering in process monitoring," *Chemometrics and Intelligent Laboratory Systems*, vol. 45, no. 1, pp. 23–38, 1999.
[6] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Computing Surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.
[7] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, 1998.
[8] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik, "Support vector clustering," *The Journal of Machine Learning Research*, vol. 2, pp. 125–137, 2002.
[9] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.
[10] D. Dong and T. J. McAvoy, "Nonlinear principal component analysis—based on principal curves and neural networks," *Computers & Chemical Engineering*, vol. 20, no. 1, pp. 65–78, 1996.
[11] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
[12] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," *Advances in Neural Information Processing Systems*, vol. 19, p. 153, 2007.
[13] Y. Bengio, "Learning deep architectures for ai," *Foundations and trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.
[14] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE Journal*, vol. 37, no. 2, pp. 233–243, 1991.
[15] ——, "Autoassociative neural networks," *Computers & Chemical Engineering*, vol. 16, no. 4, pp. 313–328, 1992.
[16] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 2, pp. 224–227, 1979.
[17] A. Cochocki and R. Unbehauen, *Neural networks for optimization and signal processing*. John Wiley & Sons, Inc., 1993.
[18] K. Bache and M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml