A Deep Clustering Algorithm based on Gaussian Mixture Model

IOP Conf. Series: Journal of Physics: Conf. Series **1302** (2019) 032012 doi:10.1088/1742-6596/1302/3/032012

A Deep Clustering Algorithm based on Gaussian Mixture Model

Xianghong Lin, Xiaofei Yang and Ying Li

College of Computer Science and Engineering, Northwest Normal University, Lanzhou 730070, China.
Email: linxh@nwnu.edu.cn

Abstract. Clustering autonomously learns the implicit cluster structure in the original data without prior knowledge. The effect of ordinary clustering algorithms is not good to cluster high-dimensional data. In this paper, we propose a deep clustering algorithm based on Gaussian mixture model, which combines two models of stacked auto-encoder and Gaussian mixture model. This algorithm uses the expectation maximization algorithm of reducing dimension data feature to train Gaussian mixture and updates the data cluster so that the data is clustered in the feature space. The experimental results demonstrate that the proposed algorithm improves the clustering accuracy, and verifies the effectiveness of the algorithm.

1. Introduction

Nowadays, with the advent of the emerging mobile Internet era and the rapid development of ecommerce technology, a large amount of real information is processed by computers to form large-scale data, such as web page information on the Internet, ultrasonic images of medical examinations, and biotechnology. Due to large scale, high real-time, complex internal data structures and potential correlations, the effect of processing these data when using traditional techniques is not good. Since deep learning is data-driven and automatically learns representative and hierarchical abstract features from sufficient training data rather than manually generated features, deep learning perform well [1, 2]. Deep learning mimics the cognitive process of human brains for feature learning rather than using traditional process of artificially extracting features [3].

In many applications of deep learning, most training models use supervised learning. But most of the data in the real world is unlabeled, and attaching label on artificially extracted features is expensive and often accompanied by human error. Therefore, unsupervised learning is expected to develop more general models to learn and experiment, and there is still huge research space in the future. Cluster analysis is a typical unsupervised learning technique in the field of machine learning. There are usually some implicit internal patterns in the raw data, and cluster analysis is a technique for exploring and discovering these implicit internal patterns. The clustering method automatically divides the data sample set and all sample points of the sample set into multiple clusters [4]. In recent years, with the continuous improvement of information science and technology, cluster analysis faces more and more problems, and the data dimension is getting higher and higher. In 2006, Torre [5] combined with dimensionality reduction and clustering, firstly clustered data with K-means, and then projected data to lower dimension of variance maximization among groups. In 2016, Xie [6] proposed deep embedding clustering which used deep neural networks to perform nonlinear feature extraction for cluster analysis.

Because of the excellent performance of deep learning in high-dimensional large-scale datasets, this paper combined deep learning with clustering and proposed a deep clustering algorithm based on

Gaussian mixture model. The algorithm transforms the original input data into abstract features through the hidden layer of the stacked auto-encoder, and then uses the Gaussian mixture model to cluster the unlabeled features. The algorithm was evaluated by experiments and the performance of the algorithm on different standard datasets was analysed.

2. Methods

2.1. Auto-Encoders

Auto-encoders are important in machine learning because they perform information preserving after dimension reduction. A single-layer auto-encoder, which is a kind of neural network consisting of only one hidden layer, sets the target values to be as equal to the input as possible. Deep neural networks use it, as an element, to find common data representation from the input [7]. The network structure of a single-layer auto-encoder is shown in Figure 1. The input x is converted to the feature *h* by the mapping function, and the feature *h* is mapped to the reconstruction data r by the other mapping function. The back propagation algorithm is used to adjust the weights and biases of the network to reduce the error value in the training process of auto-encoder.

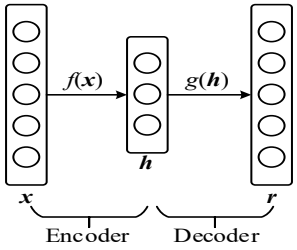

Figure 1. The structure of an auto-encoder network.

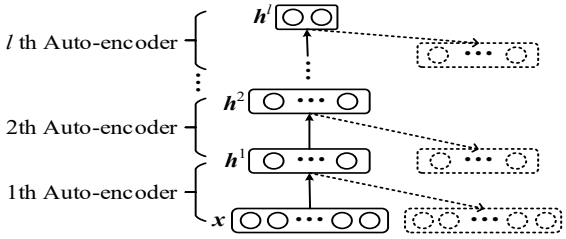

Figure 2. The network structure of a stacked auto-encoder. The representation of the (*l*-1)th hidden layer is used as input for the *l*th hidden layer.

Recent studies in machine learning has shown that a deep or hierarchical architecture is useful to find highly non-linear and complex patterns in data [8]. Inspired from the studies, in this paper, we consider a stacked auto-encoder (SAE) [9], which is composed of several auto-encoders stacked and combined. There are two training manners of SAE: bottom-up training and greedy layer-wise training. It is generally known that deep networks trained in bottom-up training manner suffers from falling into a poor local optimum [10]. In this paper, we choose the greedy layer-wise unsupervised learning algorithm. The key idea is to train one layer at a time and the latter layer input from the output of previous encoder [11]. The greedy layer-wise training is also called pre-training. The pre-training is performed in an unsupervised manner with a standard back-propagation algorithm [12]. The Figure 2 shows an overall model of SAE with multiple auto-encoders stacked hierarchically.

For an *l* layer stack auto-encoder neural network, we assume that $W(k, 1)$, $W(k, 2)$, $b(k, 1)$, $b(k, 2)$ indicates that the *k*th encoder parameters is corresponding $W(1)$, $W(2)$, $b(1)$, $b(2)$, then the encoding step of *l*th layer is defined as follows:

$$\boldsymbol{h}^{l} = \sigma\left(\boldsymbol{W}(l,1)\boldsymbol{h}^{l-1} + \boldsymbol{b}(l,1)\right) \tag{1}$$

$$\boldsymbol{r}^{l} = \sigma\left(\boldsymbol{W}(l,2)\boldsymbol{h}^{l} + \boldsymbol{b}(l,2)\right) \tag{2}$$

The training optimization is performed by minimizing the mean square error in this paper.

2.2. Gaussian Mixture Model

Gaussian mixture model (GMM) is a widely used clustering algorithm, which is a parametric probability density function represented as a weighted sum of Gaussian component densities [13]. GMM parameters are estimated from training data using the iterative Expectation-Maximization (EM)

algorithm estimation from a well-trained prior model. For a mixture model, it is assumed that a given sample **x** is the realization of a random vector which distribution is a mixture (convex combination) of several class conditioned distributions [14]:

$$P(\boldsymbol{x}\mid\theta)=\sum_{k=1}^{K}\alpha_k\phi(\boldsymbol{x}\mid\theta_k) \tag{3}$$

where **x** represents a D-dimension continuous data vector (i.e. features), *K* is the number of the models, α_k is the probability that the *k*th class in the sample set is selected and $\sum_{k=1}^{K}\alpha_k$. $\phi(x\mid\theta_k)$ denotes the Gaussian distribution density of the *k*th Gaussian model component, and its function is expressed as

$$\phi(\boldsymbol{x}\mid\theta_k)=\frac{1}{\sqrt{2\pi}\sigma_k}\exp\left(-\frac{(\boldsymbol{x}-\mu_k)^2}{2\sigma_k^2}\right) \tag{4}$$

where the mean value is μ_k, the variance is σ_k^2. A complete Gaussian mixture model is parameterized by averaging the mean, the variance, and the mixed weights of all component models and these parameters are expressed as

$$\theta=\left\{\mu_k,\delta_k^2,\alpha_k\right\},(k=1,...,K) \tag{5}$$

Given the training data vector and the configuration of the GMM, the expectation maximization algorithm is used to iteratively estimate the parameters of the GMM so that the trained eigenvectors closely distributed in a sense.

For N training vectors $\boldsymbol{X}=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_N\}$, assuming that each vector is independent of each other, the GMM likelihood can be written as

$$p(\boldsymbol{X}\mid\theta)=\prod_{n=1}^{N}p(\boldsymbol{x}_n\mid\theta) \tag{6}$$

Take the logarithm of the GMM likelihood function and maximize its logarithm likelihood function as follows

$$\max_{\theta}\ln p(\boldsymbol{X}\mid\theta)=\sum_{n=1}^{N}\ln\sum_{k=1}^{K}\alpha_k\phi(\boldsymbol{x}\mid\theta_k) \tag{7}$$

The above non-linear functions with respect to parameter θ obtain a maximized expectation in a special case by using iterative maximization likelihood parameter estimation, rather than maximize parameters directly [15]. The basic idea of the EM algorithm is estimate a new parameter from the initial parameter, making $p(\boldsymbol{X}\mid\bar{\theta})\geq p(\boldsymbol{X}\mid\theta)$. Then, the new parameter is used as the initial parameter for the next iteration, and this process is repeated until a certain convergence threshold.

The number of clusters *K* is set in advance in the process of the clustering, and variable $\Pr(k\mid\boldsymbol{x}_n,\theta)$ which is introduced represents the probability of the *n*th training data from the *k*th model. The posterior probability is calculated as follows:

$$\Pr(k\mid\boldsymbol{x}_n,\theta)=\frac{\alpha_k\phi(\boldsymbol{x}\mid\theta_k)}{\sum_{n=1}^{K}\alpha_k\phi(\boldsymbol{x}\mid\theta_k)} \tag{8}$$

In each EM iteration, the following reestimate formulas are used to ensure the monotonous increase of the likelihood value of the parameter

$$\bar{\alpha}_k=\frac{1}{N}\sum_{n=1}^{N}\Pr(k\mid\boldsymbol{x}_n,\theta) \tag{9}$$

$$\overline{\mu}_k = \frac{\sum_{n=1}^{N} \Pr(k \mid \boldsymbol{x}_n, \theta)\boldsymbol{x}_n}{\sum_{n=1}^{N} \Pr(k \mid \boldsymbol{x}_n, \theta)} \tag{10}$$

$$\overline{\sigma}_k^2 = \frac{\sum_{n=1}^{N} \Pr(k \mid \boldsymbol{x}_n, \theta)\boldsymbol{x}_n^2}{\sum_{n=1}^{N} \Pr(k \mid \boldsymbol{x}_n, \theta)} - \overline{\mu}_k^2 \tag{11}$$

where, μ_k, σ_k^2 and x_n refer to any of the elements $\boldsymbol{\mu}_k$, $\boldsymbol{\sigma}_k^2$ and $\boldsymbol{x}_n$, respectively.

3. A DEEP CLUSTERING ALGORITHM BASED ON GMM

It is a challenging problem to cluster common data that the original space contains various complex structures. The traditional approach solves this problem by adding feature learning methods that either capture the intrinsic structure of the data or represent the data for better clustering. The deep clustering algorithm uses neural network learning to facilitate deep feature representation of clustering tasks. In this subsection, we use the combination of the stacked auto-encoder and the Gaussian mixture model to train the GMM and update the data cluster using the expectation maximization algorithm for the extracted encoded data. The structure diagram of deep clustering algorithm based on GMM is shown in Figure 3.

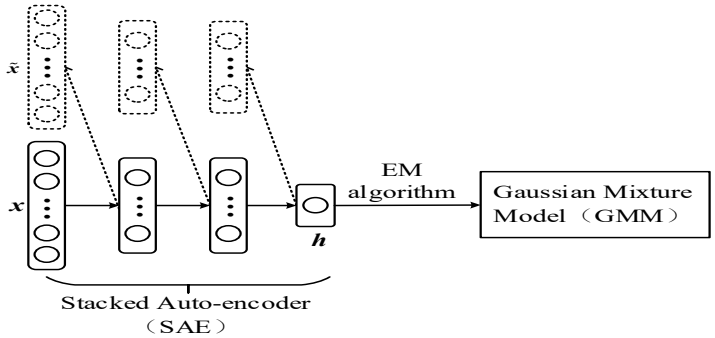

Figure 3. The structure of a deep clustering algorithm based on the GMM.

In this deep clustering algorithm, every sample of dataset $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}$ is D-dimension. Firstly, we set up a stacked auto-encoder based on a neural network consisting of encoder E^l and decoder $D^l(l = 0, \cdots, L)$, and parameters of the SAE are θ_D^l and θ_E^l, respectively, where l represents the layer number of stacked auto-encoders. The encoder maps input data from the original D-dimension data space to a d-dimensional space ($d < D$).The decoder D^l maps feature data from the encoding space to the original D-dimension data space, making each data point $\boldsymbol{x}_n (n = 1, \cdots, N)$ roughly equals $D(E(\boldsymbol{x}_n)) \approx \boldsymbol{x}_n$ by reconstructing the encoder and decoder. If the reconstruction is feasible, the auto-encoder will compress the information of each sample without causing too much damage. Secondly, after training the auto-encoder of the first layer, the coding of the first layer of the auto-encoder is used as the input data of the second layer auto-encoder, and then the second layer auto-encoder is trained, which makes the loss of the auto-encoder of this layer as small as possible. Finally, after training the L layer automatic encoders, the encoded data of last layer is taken as the feature $\boldsymbol{H} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_N\}$, so that the feature extraction stage is completed. In this process, the unified combination of each auto-encoder is called a stacked auto-encoders. In the feature extraction stage, each auto-encoder is trained with greedy layer-wise unsupervised learning way.

After training in the feature extraction stage, the feature space data learned by stacked auto-encoder are sent to the next stage clustering algorithm for clustering. In this paper, Gaussian mixture model is

used as the basic clustering algorithm, which is used to estimate the probability distribution of the given data. The overall distribution is composed of several components, where the number of clusters K is defined beforehand (knowing the number of Gaussian distributions K). $\theta = \{\theta_1, \theta_2, \cdots, \theta_K\}$ is defined as the parameter set of Gaussian mixture model, where $\theta_k = (\mu_k, \sigma_k^2)(k = 1, \cdots, K)$ represents the parameter of the kth Gaussian mixture component. We use the expectation maximization algorithm to estimate each θ_k , that is to say, each data can be expressed by its probability. If the probability of calculation is greater, the contribution will be greater, and vice versa. The weighted mean and variance are calculated by each sample's contribution to the Gaussian distribution as weight, and then the original parameters of the Gaussian distribution are updated until the algorithm converges to the local optimal or the maximum number of iterations. A detailed description of the deep clustering process based on the GMM network is described as Algorithm 1.

Algorithm 1: Deep clustering process based on the GMM

Input: $\boldsymbol{X}=\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}$, Cluster number K, Sample number N

Output: Divide X into k group $\{C_1, C_2, \cdots, C_k\}$, where $\cup_{i=1}^{k} C_i = \boldsymbol{X}$

1. Initialize the parameters θ_D^l and θ_E^l of auto-encoder for each layer
2. **for l ← 1 to L do**
3. Optimization of reconfiguration error of automatic encoder θ_D^l and θ_E^l in lth layer
   $L(\theta_E^l, \theta_D^l) = \sum_{n=1}^{N} \left\| \boldsymbol{h}_n^{l-1} - D(E(\boldsymbol{h}_n^{l-1})) \right\|^2, \ (\boldsymbol{h}_n^0 = \boldsymbol{x}_n)$
4. Update the parameters θ_D^l and θ_E^l of the automatic encoder for lth layer
5. **end for**
6. Initialization of the parameters of Gaussian model $\theta = \{\mu_k, \sigma_k^2, \alpha_k\}(k = 1, \cdots, K)$
7. **repeat**
8. **for n ← 1 to N do**
9. **for k ← 1 to K do**
10. Calculate the probability of nth training data $\boldsymbol{x}_n$ from the k model
11. **end for**
12. **end for**
13. **for k ← 1 to K do**
14. Calculating the parameters of the kth Gaussian model $\mu_k, \sigma_k^2, \alpha_k$
15. Update the parameters $\mu_k, \sigma_k^2, \alpha_k$
16. **end for**
17. **until** the cessation condition is satisfied

4. Results

4.1. Data Description and Evaluation Criteria

We select 3 datasets in UCI database and 2 image datasets (COIL-20, MNIST) to evaluate the proposed algorithm. The detailed information of all the data in the experiment is listed in Table 1, containing the samples number, property and the number of classes.

Table 1. Detailed introduction of experimental data

Dataset	Samples number	Property	Samples number
Iris	150	4	3
Wine	178	13	3
Isolet	7797	617	26
COIL-20	1440	1024	20
MNIST	70000	784	10

We use the standard unsupervised evaluation criterion to evaluate and compare the clustering algorithms [16]. For all algorithms, we set the clusters number as the real labels number of data, and evaluate their performance with unsupervised clustering accuracy (ACC)

$$ACC = \max_{m} \frac{\sum_{i=1}^{n} \mathbf{1}\{l_i = m(c_i)\}}{N} \tag{12}$$

where N is the total number of data samples, l_i is the real label of the ith data, c_i is the prediction value of the ith data generated by the clustering algorithm, and m is the mapping range of all possible pairs between the clusters and labels. ACC reflects whether the clustering algorithm correctly discovers the category structure of the original data.

4.2. Experimental Parameters

For all the algorithm involved in the experiments, we repeat the 10 random experiments to the average value as the final result of the experiment. In the experimental dataset, the SAE network structure is set up 100-50-10 for the first two datasets (Iris and Wine), *D*-500-300-30 for the latter three datasets, where *D* is the dimension of data space and the neurons among all layers are all connected. For stacked auto-encoders, we set that the weights is initialized to the Gaussian distribution with zero mean and standard deviation of 0.01, the number of iterations is 10000, the mini-batch size is 100, and the learning rate is 0.001. In the GMM clustering stage, the maximum number of iterations for setting GMM is 300.

4.3. Experimental Analysis and Comparison

4.3.1. Experimental comparison of feature extraction step. Figure 4 shows the reconstruction error curve of each layer AE on the MNIST dataset during the training and the test process of a stacked auto-encoder. It can be seen from the graph that the reconstruction error tends to decrease with increase of epochs, and it begins to descend rapidly, then slow down until reaches a steady state. Moreover, with the increase of the number of encoder layers, the overall error trend is smaller, and the training set is smaller than the test set.

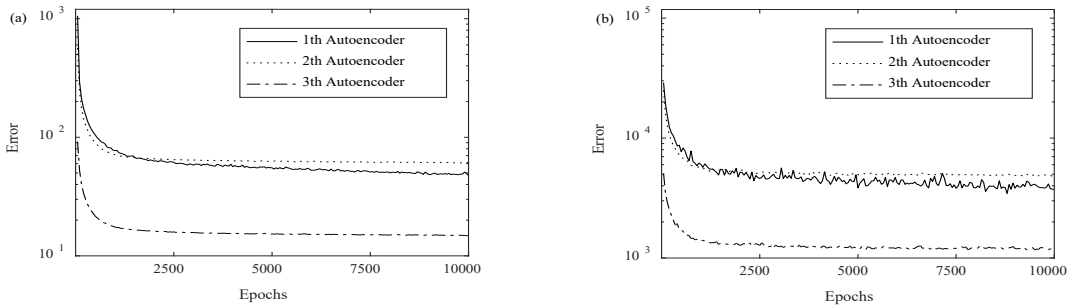

Figure 4. The change curve of the reconstruction error of each layer of a SAE on the MNIST dataset. (a) The curve of the training set; (b) The change of the test set.

Figure 5 shows the change curve of the clustering accuracy that the MNIST dataset is coded as different number of feature. In order to test the influence of the features number on the clustering accuracy, we change the number of neurons of the last layer and the other layers in the SAE is constant. The network structure is set to 784-500-300-*H*, and *H* represents the number of features (the variables) that are encoded. From the graph we can see that the value of the ACC increases at first, then decreases when the feature number increases gradually, so the suitable feature number for the SAE is 30.

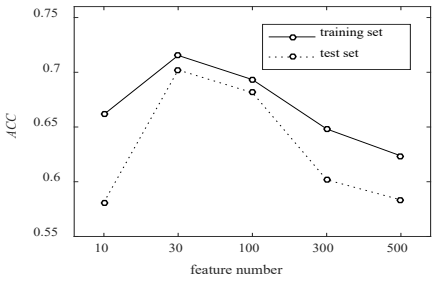

Figure 5. Comparison of clustering accuracy of different number of features.

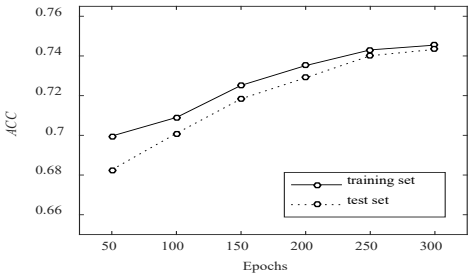

Figure 7. The relationship between the iterations in MNIST dataset and the accuracy of clustering.

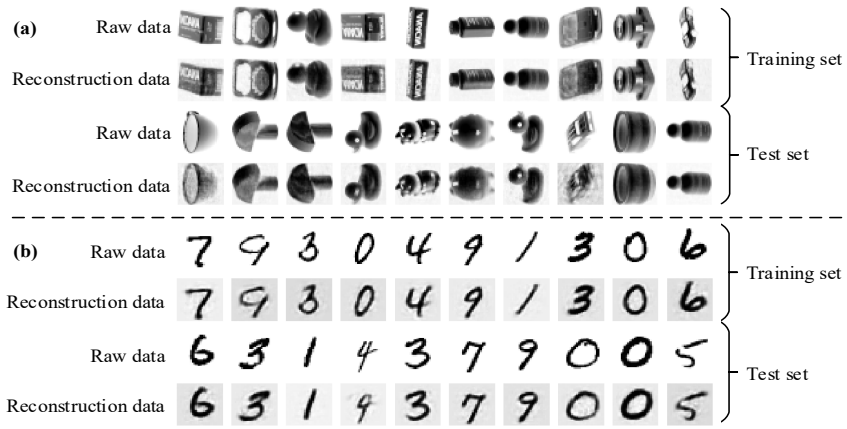

Figure 6. A partial contrast diagram of reconstruction data and the original data. (a) The contrast diagram of MNIST dataset before and after reconstruction. (b) The contrast diagram of COIL-20 dataset before and after reconstruction.

Figure 6 shows the contrast of the raw and reconstructed data after the SAE dimensionality reduction on two datasets, where reconstructed data is obtained by the reconstruction of 30 features after the SAE dimensionality reduction. It can be seen that there is a certain difference between the original and reconstructed pictures, but the information of reconstructed pictures is clear.

4.3.2. Experimental comparison of clustering step. Figure 7 shows the relationship between the iterations of the training set and the test set of the MNIST dataset and the clustering accuracy in the training process. As can be seen from the graph, the accuracy increases with the increase of the iterations, and the accuracy of clustering reaches a stable state after the number of iterations is *250*.

Table 2 describes the comparison of clustering accuracy between GMM and the proposed algorithm on 5 different datasets. It can be seen from the table that the proposed algorithm has a certain increase in clustering accuracy compared with the basic algorithm GMM, and the increase of 5 datasets on the training set is 6.16%, 2.93%, 3.45%, 7.39% and 20.67% respectively.

Table 2. Comparison of clustering accuracy of different clustering algorithms.

Dataset	GMM		Proposed method	
	Training set	Test set	Training set	Test set
Iris	90.89%	90.0%	97.05%	96.58%
Wine	92.63%	94.0%	95.56%	96.44%
Isolet	57.97%	38.2%	61.42%	62.32%
COIL-20	67.96%	69.9%	75.35%	74.64%
MNIST	53.88%	53.15%	74.55%	74.33%

Figure 8. Partial effect graph of MNIST dataset clustering.

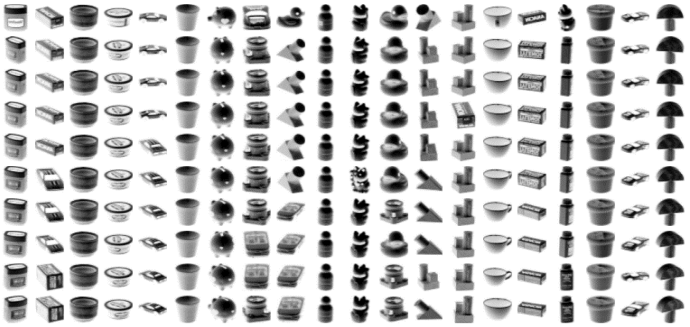

Figure 9. Partial effect graph of COIL-20 dataset clustering.

Figure 8 and Figure 9 show the clustering effects of random sampling MNIST and COIL-20 data. From Figure 8, it can be seen that the algorithm has a good clustering effect on some categories of MNIST datasets, such as '2' and '0', and there is no obvious distinction between some categories (such as '7' and '9', '3' and '5'). From Figure 9, the clustering visualization effect of the algorithm on the COIL-20 dataset is obvious. There are only a few types of pictures that are not made a distinction.

5. Conclusions

In this paper, we propose a deep clustering algorithm based on Gaussian mixture model, which combines stacked auto-encoder and Gaussian mixture model. In the feature extraction stage, we choose the layer-wise unsupervised method to greedily train the stacked auto-encoder, so that obtaining high-dimensional abstract feature data of the input data. Then, the learned feature space data is sent to a Gaussian mixture model for clustering, and the expected maximum algorithm is used in the clustering process to estimate the parameters of the Gaussian mixture component. The contribution of each sample to the Gaussian distribution is used as a weight to calculate the weighted mean and variance, and then the original parameters of the Gaussian distribution are updated until the algorithm converges to the local optimal solution or reaches the maximum number of iterations. The clustering performance of the proposed algorithm and Gaussian mixture model on five different datasets is compared by experimental analysis, and the effectiveness of the algorithm is proved.

6. Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant No. 61762080, and the Program for Innovative Research Team in Northwest Normal University under Grant No. 6008-01602.

7. References

[1] Yi-Ou L, Hang L and Xiao-Yu L I et al 2017 Deep learning in NLP: methods and applications *Journal of University of Electronic Science & Technology of China* **46(6)** pp 913-19
[2] Gheisari M, Wang G and Bhuiyan M Z A 2017 A survey on deep learning in big data *IEEEInternational Conference on Computational Science & Engineering*
[3] Lecun Y, Bengio Y and Hinton G 2015 Deep learning *Nature* **521(7553)** pp 436-44
[4] Jain A K, Murty M N and Flynn P J 1999 Data clustering: a review *Acm Computing Surveys* **31(3)** pp 264-323
[5] Torre F D L and Kanade T 2006 Discriminative cluster analysis *In Theory and Novel Applications of Machine Learning*
[6] Xie J, Girshick R and Farhadi A 2015 Unsupervised deep embedding for clustering analysis *Computer Science*
[7] Ngiam J, Khosla A and Kim M et al 2011 Multimodal deep learning *Proceedings of the 28th international conference on machine learning (ICML-11)* pp 689-96
[8] Bengio Y 2009 Learning Deep Architectures for AI *Foundations and Trends® in Machine Learning* **2(1)** pp 1-127

[9] Bengio Y, Lamblin P and Popovici D et al 2007 Greedy layer-wise training of deep networks *Advances in neural information processing systems* pp 153-60
[10] Larochelle H, Bengio Y and Louradour J et al 2009 Exploring strategies for training deep neural networks *Journal of machine learning research* **10(Jan)** pp 1-40
[11] Hinton G E, Osindero S and The Y W 2006 A fast learning algorithm for deep belief nets *Neural computation* **18(7)** pp 1527-54
[12] Bishop C M 1995 Neural networks for pattern recognition *Agricultural Engineering International the Cigr Journal of Scientific Research & Development Manuscript Pm* **12(5)**
[13] Mclachlan G and Mclachlan G 2004 In Advanced algorithmic approaches to medical image segmentation *Finite Mixture Models (Springer-Verlag New York, Inc)*
[14] Dilokthanakul N, Mediano P A M and Garnelo M et al 2016 Deep unsupervised clustering with gaussian mixture variational autoencoders *J*
[15] Fraley C and Raftery A E 2002 Model-based clustering, discriminant analysis, and density estimation *Journal of the American statistical Association* **97(458)** pp 611-31
[16] Yang Y, Xu D and Nie F et al 2010 Image clustering using local discriminant models and global integration *IEEE Transactions on Image Processing* **19(10)** pp 2761-73