Information Sciences 424 (2018) 27–38

Contents lists available at ScienceDirect

Information Sciences

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

Subspace clustering using a low-rank constrained autoencoder

Yuanyuan Chen, Lei Zhang, Zhang Yi*

Machine Intelligence Laboratory, College of Computer Science, Sichuan University, Chengdu 610065, PR China

ARTICLE INFO

Article history:
Received 28 March 2017
Revised 14 September 2017
Accepted 16 September 2017
Available online 18 September 2017

Keywords:
Deep neural networks
Subspace clustering
Autoencoder
Low-rank representation

ABSTRACT

The performance of subspace clustering is affected by data representation. Data representation for subspace clustering maps data from the original space into another space with the property of better separability. Many data representation methods have been developed in recent years. Typical among them are low-rank representation (LRR) and an autoencoder. LRR is a linear representation method that captures the global structure of data with low-rank constraint. Alternatively, an autoencoder nonlinearly maps data into a latent space using a neural network by minimizing the difference between the reconstruction and input. To combine the advantages of an LRR (globality) and autoencoder (self-supervision based locality), we propose a novel data representation method for subspace clustering. The proposed method, called low-rank constrained autoencoder (LRAE), forces the latent representation of the neural network to be of low rank, and the low-rank constraint is computed as a prior from the input space. One major advantage of the LRAE is that the learned data representation not only maintains the local features of the data, but also preserves the underlying low-rank global structure. Extensive experiments on several datasets for subspace clustering were conducted. They demonstrated that the proposed LRAE substantially outperformed state-of-the-art subspace clustering methods.

1. Introduction

Subspace clustering seeks to determine clusters in different subspaces within a dataset, and has been widely used in many scientific and industrial applications, including the clustering of images containing different objects, segmentation of different objects in a video sequence, and separation of a mixture of speech. Formally, let $X = [x_1, x_2, \cdots, x_N]$ be a $d \times N$ data matrix that consists of N vectors that are assumed to belong to K clusters. Subspace clustering determines to which cluster each datum is assigned. Among different subspace clustering methods, spectral clustering is an important branch for managing high-dimensional data.

One key step in the spectral clustering algorithm is to construct an affinity matrix $C \in R^{N \times N}$, where C_{jk} quantizes the similarity between data points j and k. A popular measurement of similarity is a function of some distance. For example, in [21], the similarity between two points is defined by $C_{jk} = \exp\left\{-s_{jk}^2\right\}$, where s_{jk} is a value of the distance between j and k. After obtaining the affinity matrix C, the segmentation of the data is performed by applying the k-means algorithm to the largest k eigenvectors of a normalized Laplacian matrix.

Different from [21], ℓ_1-graph [4], L2-graph [25], and sparse subspace clustering (SSC) [6], low-rank representation (LRR) [12,13,16,17] learns a representation by constructing the affinity matrix using the linear reconstruction coefficient. The major

* Corresponding author.
E-mail address: zhangyi@scu.edu.cn (Z. Yi).

http://dx.doi.org/10.1016/j.ins.2017.09.047
0020-0255/© 2017 Elsevier Inc. All rights reserved.

difference between the approaches is the constraint on the coefficient, which describes the prior on the data distribution. Although these methods are state of the art in the area of subspace clustering, we observe that they suffer from the following limitations: First, these methods have to use the entire dataset as a dictionary; thus, they cannot manage large scale and incremental data [26]. Second, these methods assume that the data can be linearly represented with respect to each other. In this case, subspace clustering could be solved properly using convex methods. However, in reality, the relations between data are usually highly nonlinear or "nonlinear in linear" [11,14,23,27]. The aforementioned methods might achieve degraded performance. Third, spectral clustering based approaches inherit the disadvantages of manifold learning because they can be the same as performing manifold learning to obtain a data representation and conducting k-means clustering on the representation to achieve clustering membership. The disadvantages of manifold learning, such as poor robustness and a smooth sampling requirement, further limit applications of subspace clustering methods in the scenario of big data.

As another popular data representation method, a deep autoencoder [9] overcomes the disadvantages of manifold learning. A deep autoencoder progressively maps input data into a latent space by minimizing the difference between the reconstruction and input using a parametric neural network. Unlike manifold learning, an autoencoder performs in parallel and has a fast inference speed, which enables it to manage large scale and incremental data. It is a data-driven approach and does not depend on the linear assumption. Furthermore, the deep architecture [22] can capture the nonlinear complex structure of big data because of the nonlinear activation function and multi-layered structure.

Motivated by the substantial success of deep learning [9,20,33,37], we propose a novel subspace clustering method that combines the advantages of traditional LRR models and the popular deep model. The proposed method, called a low-rank constrained autoencoder (LRAE), is a variant of an autoencoder. Unlike the standard autoencoder and LRR, the LRAE not only enforces the latent representation that can reconstruct input well, but also requires the set of latent representations to be of low rank, where the low-rank constraint is computed from the input space. To summarize, the LRAE integrates the locality of an autoencoder and globality of LRR to learn a good data representation, and thus achieve better clustering results. To the best of our knowledge, this is the first work to incorporate low-rank constraint into an autoencoder. The most closely related work to our proposed method PARTY which was proposed in [24]. However, the LRAE and PARTY are quite different. In details, PARTY assumes that the latent representation can be sparsely reconstructed, whereas the LRAE adopts a low-rank prior. In practice, a low-rank prior is better than a sparsity prior because the former can guarantee the globality of the entire dataset, whereas the latter cannot. Moreover, the objective functions of PARTY and the LRAE are different. PARTY aims to solve an ℓ_1-minimization problem to achieve sparsity, whereas our method achieves a low-rank with a nuclear-norm based constraint. Benefiting from the intrinsic property of our objective function, our method is much more efficient than PARTY.

The remainder of this paper is organized as follows: In Section 2, we briefly discuss related work. In Section 3, we present the LRAE in detail, including the network architecture, objective function, and learning algorithm. We report the results of experiments and make some discussion in Section 4. We present our conclusions in Section 5.

2. Related work

In this section, we briefly discuss existing work on LRR and autoencoders.

2.1. Low-rank representation

In [13], Liu et al. proposed an LRR to determine the coefficient matrix C of data X with the lowest rank:

$$\min_{C} rank(C) \quad \text{s.t.} \quad X = AC, \tag{1}$$

where A is a dictionary of the data. If the data in X are arranged to satisfy the true segmentation of the data, the optimal solution to (1), C^*, is block diagonal.

$$C^* = \begin{bmatrix} C_1^* & 0 & \cdots & 0 \\ 0 & C_2^* & \cdots & 0 \\ 0 & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & C_K^* \end{bmatrix}, \tag{2}$$

which captures both dense within-class affinities and zero between-class affinities. Compared with sparse representation, low-rank representation of data captures the global structure of the data. In [14,15], Liu established a deeper theoretical analysis of LRR. Based on Liu's main idea, a great number of researchers have attempted to search a low-rank matrix to reveal the structure of the data or improve the efficiency of the original algorithm [3,7,18,32].

By solving a nuclear norm minimization problem, Favaro [7] proposed a closed-form solution to the low-rank minimization problem:

$$\min_{C} \|C\|_* + \frac{\lambda}{2}\|X - XC\|_F^2. \tag{3}$$

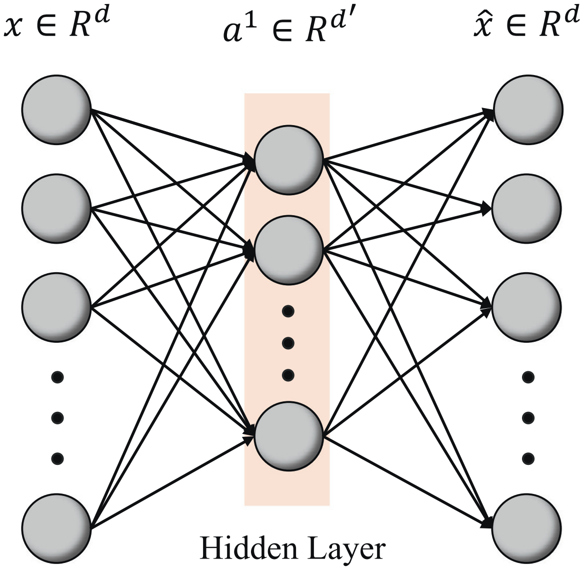


Fig. 1. A basic autoencoder architecture with one hidden layer.

Lu et al. proposed a robust and efficient subspace segmentation algorithm [18] using least square regression (LSR). By applying the Frobenius norm, LSR solves the following optimization problem:

$$\min_C \|X - XC\|_F^2 + \frac{\lambda}{2}\|C\|_F^2, \tag{4}$$

which has an analytical solution.

Low-rank based algorithms have demonstrated promising performance for various machine learning applications, such as image classification, image clustering, and motion segmentation.

2.2. Autoencoder

Autoencoders were first proposed by Hinton and Rumelhart et al. [28] to solve the problem of “backpropagation without a teacher”. As the name suggests, autoencoders are data-specific, and learn automatically from datasets rather than being engineered by a human. Autoencoders aim to encode the input data into a representation and then decode the representation to outputs with the least loss. Because of their simple concept, they play a fundamental role in unsupervised learning and other tasks. Recently, combined with the “deep architecture” approach, autoencoders have been generally stacked and trained in an unsupervised manner followed by a supervised output layer [9]. Because of their advantages, autoencoders have been applied to solve a number of real-world problems, such as bilingual learning [30], HIV analysis [1], and speech motion recognition [5]. A basic autoencoder architecture with one hidden layer is shown in Fig. 1. Given parameters W_1, b_1, W_2, and b_2 and activation functions $f_1(\cdot)$ and $f_2(\cdot)$, the network takes an input $x \in R^d$, and maps it onto the last layer :

$$a^{(2)} = f_2(W_2 \cdot f_1(W_1 \cdot x + b_1) + b_2), \tag{5}$$

where $a^{(2)} \in R^d$ is the activation in the last layer.

Alternatively we can use a single function $h(\cdot)$ to represent the mapping from the input data to the activation of neurons in the last layer.

Unlike a general neural network with three layers, an autoencoder maps input data x onto a reconstruction of itself, that is, $\hat{x}$, which has the same dimension as x. From another point of view, an autoencoder attempts to learn an approximation to the identity function, to make $\hat{x}$ equal to x.

Formally, for an input dataset X that has N samples, a general autoencoder problem minimize the reconstruction error with a regularization term:

$$J(W^{(m)}, b^{(m)}) = \mathcal{D}(X, \hat{X}) + \lambda \mathcal{R}(W^{(m)}, b^{(m)}), \tag{6}$$

where $W^{(m)}$ and $b^{(m)}$ are the weights and biases of the m-th layer in the network, and

$$\mathcal{D}(X, \hat{X}) = \left\|X - \hat{X}\right\|_F^2. \tag{7}$$

From the angle of probability, considering the regularization term, it is a priori distribution of the mapping function $h(\dot{})$. For example, the sparse autoencoder attempts to minimize the following cost function:

$$J(W,b) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}\left\|(a_i^{(2)}) - x_i\right\|^2 + \frac{\lambda}{2}\sum_{m=1}^{2}\sum_{i=1}^{n^{(m)}}\sum_{j=1}^{n^{(m-1)}}\left(W_{ij}^{(m)}\right). \tag{8}$$

The second item in Eq. (8) limits the magnitude of the weights, which may prevent overfitting.

Based on this framework, various types of autoencoders can be derived according to different constraints on the cost function, activation functions, number of layers, and learning algorithms. For example, the sparse autoencoder [20] adds a Kullback–Leibler divergence term as a penalty to constrain the magnitude of the hidden neurons' activations. The hidden units activations must mostly be near zero. Vincent et al. proposed the stacked denoising autoencoder [33], which was already used in image processing and other applications; in particular, it was applied to extract noisy robust features for classification. The stacked sparse autoencoder [35] was presented for efficient nuclei detection on high-resolution histopathological images of breast cancer. In most actual applications, autoencoders are stacked and trained in an unsupervised manner, followed by a supervised learning phase to train the output layer and fine-tune the weights in the entire network. These deep architectures have been shown to achieve state-of-the-art results on a number of classification and regression problems.

3. Low-rank constrained deep autoencoder for subspace clustering

In this section, we describe the details of the proposed LRAE algorithm for subspace clustering, including the network architecture, loss function, and learning algorithm. Then, we propose the entire algorithm that applies the LRAE for subspace clustering. Additionally, we analyze the relationship between the proposed LRAE and related work.

3.1. Network architecture

As shown in Fig. 2, the proposed LRAE is a multi-layer neural network. Suppose there are $M+1$ layers in total, where M is an even number; that is, layer $M/2$ is the middle layer, which is represented by the subscript "Mid". Therefore, M groups of weights and biases need to be calculated in a neural network, which are denoted by $W^{(1)}, \cdots, W^{(M)}, b^{(1)}, \cdots, b^{(M)}$. $W_{ij}^{(m)}$ denotes the parameter associated with the connection between neuron j in layer $m-1$ and neuron i in layer m. The leftmost layer is the input layer. The input data for the autoencoder is represented by $a^0 = x$. The activation (or output) of the ith neuron in layer $m+1$ is denoted by $a^{(m)}$. Accordingly, for the entire dataset, the activations in these $M+1$ layers are represented by $A^0, \cdots, A^{(m)}, \cdots, A^{(M)}$. The rightmost layer is the output layer. There are $n^{(m)}$ neurons in the mth layer.

Generally, an autoencoder framework is considered as a combination of encoding and decoding processes. Therefore, the network encodes the inputs to the representation in the hidden layers through $(M-1)/2$ encoding processes and then decodes the hidden layers to obtain the reconstruction of the input. More generally, the neurons' activation in each layer can

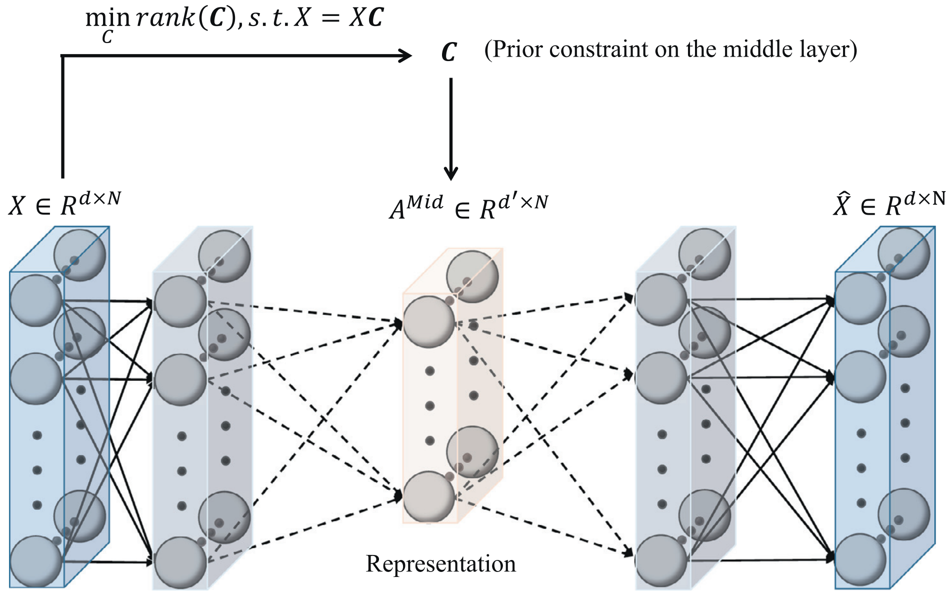

$\min_{C} rank(\boldsymbol{C}), s.t. X = X\boldsymbol{C}$
$\boldsymbol{C}$ (Prior constraint on the middle layer)
$X \in R^{d\times N}$
$A^{Mid} \in R^{d'\times N}$
$\hat{X} \in R^{d\times N}$
Representation

Fig. 2. A basic autoencoder architecture with one hidden layer.

be considered as a representation of the original input and thus for subspace clustering. In this paper, the representation in the middle layer, that is, $A^{(\mathrm{Mid})} = \left[a^{(\mathrm{Mid})}(1), \cdots, a^{(\mathrm{Mid})}(N)\right]$, is considered as the underlying representation of all N input data. The representations are used for subspace clustering.

For a single input datum x, the computation process of the network is given by

$$\begin{cases} a^{(0)} = x, \\ z_i^{(m+1)} = \sum_{j=1}^{n^{(m)}} W_{ij}^{(m)} \cdot a_j^{(m)} + b_i^{(m)}, \quad \text{for } m = 1, 2, \cdots, M. \\ a^{(m)} = g^{(m)}\left(z^{(m)}\right), \end{cases} \tag{9}$$

where $g^{(m)}(\cdot)$ is the activation function in layer m for a neuron. In this paper, we use the sigmoid function as the activation function in each layer. Therefore,

$$g(s) = \frac{1}{1+e^{-s}}. \tag{10}$$

This function is easy to differentiate because $\dot{g}(s) = g(s) \cdot (1 - g(s))$.

The second equation in (9) can be written in matrix form as:

$$z^{(m+1)} = W^{(m)} \cdot a^{(m)} + b^{(m)}. \tag{11}$$

Considering all the input data X, we rewrite Eq. (9) as a compact form:

$$\begin{cases} A^{(0)} = X; \\ Z^{(m+1)} = W^{(m)} \cdot A^{(m)} + B^{(m)}; \quad \text{for } m = 1, 2, \cdots, M, \\ A^{(m)} = g^{(m)}\left(Z^{(m)}\right), \end{cases} \tag{12}$$

which is a feedforward process that is the same as other multi-layer neural networks. To allow the network to learn optimal parameters, it is necessary to use a cost function to indicate the performance of the network.

3.2. Prior constraint

Similar to a deep autoencoder, for a single input datum, the loss function $L(x, \hat{x})$ ($L\left(X, \hat{X}\right)$ for all inputs) of the LRAE is used to optimize the parameters in the neural network model, where $\hat{x}$ is the activation of the output layer. This function seems to be a particularly trivial function to be attempting to learn; however, by placing constraints on the network, such as limiting the number of hidden units or restricting the magnitude of the weights, we can discover interesting structure about the data.

Compared with a typical autoencoder, the proposed LRAE places more constraints on the network. It not only attempts to minimize the reconstruction error via unsupervised learning, that is, $L\left(X, \hat{X}\right)$, but also tends to minimize the reconstruction error in the middle layer using other data under a prior low-rank constraint. In [24], PARTY incorporates a prior sparsity constraint on the middle layer to preserve the sparse reconstruction relation over the entire dataset. The prior sparsity constraint in PARTY aims to calculate the sparest representation of each datum individually. By contrast, the prior low-rank constraint in LRR attempts to calculate the lowest rank representation of all data jointly. Therefore, compared with PARTY, the LRAE seems to be more appropriate for subspace clustering.

3.3. Cost function

Formally, the network attempts to minimize the following objective function:

$$\min_{W^{(m)}, b^{(m)}} \frac{1}{2}\left\|X - \hat{X}\right\|_F^2 + \frac{\lambda_1}{2}\left\|A^{(\mathrm{Mid})} - A^{(\mathrm{Mid})}C\right\|_F^2 + \frac{\lambda_2}{2}\left(\left\|W^{(m)}\right\|_F^2 + \left\|b^{(m)}\right\|_2^2\right), \tag{13}$$

for $m = 1, 2, \cdots, M$, where C is the prior constraint obtained by the LRR algorithm, and λ_1 and λ_2 are parameters that indicate the influence of the two regularization terms. In the above objective function, the first term aims to minimize the reconstruction error. The second term does not constrain the weights W directly, but uses a prior low-rank matrix to constrain the output in the middle layer. Additionally, the third term restricts the magnitude of the weights in the network.

In more detail, we write the overall cost function of Eq. (13) in scalar form:

$$J = \frac{1}{2}\sum_{i=1}^{N}\left(\left\|x_i - a_i^{(M)}\right\|_2^2 + \lambda_1\left\|a_i^{(\mathrm{Mid})} - A^{(\mathrm{Mid})}c_i\right\|_2^2\right) + \frac{\lambda_2}{2}\sum_{m=1}^{M-1}\left(\left\|W^{(m)}\right\|_F^2 + \left\|b^{(m)}\right\|_2^2\right). \tag{14}$$

From Eq. (14), the definition of the first two terms in the equation becomes clearer. The first term is a local constraint on each datum. In the second term, because A is the activation for all input data, it is a global constraint that captures the underlying structure of the data.

To solve the problem in Eq. (13), that is, to update the parameters $W^{(m)}$ and $b^{(m)}$ in the network, we apply stochastic sub-gradient descent to calculate the sub-gradients of J with respect to all $W^{(m)}$ and $b^{(m)}$ in the network, that is, calculate $\partial J/\partial W^{(m)}$ and $\partial J/\partial b^{(m)}$, where $m = 1, 2, \cdots, M$.

3.4. Learning algorithm

The LRAE uses the backpropagation algorithm to learn the parameters in the neural network. To train the network, all the weights $W^{(m)}$ and biases $b^{(m)}$ should be initialized before training. Because we use stochastic gradient descent to optimize the network, the solution to the algorithm depends on the initial values of the parameters. Compared with the random initialization of deep neural networks, Glorot and Bengio [8] proposed a new initialization scheme called normalized initialization:

$$W^{(m)} \sim N\left(-\sqrt{\frac{6}{n^{(m)}+n^{(m-1)}}}, +\sqrt{\frac{6}{n^{(m)}+n^{(m-1)}}}\right). \tag{15}$$

Another initialization scheme that was proposed by Hinton [9], that is, pretraining the network with a fine-tune strategy, also demonstrates good performance.

After initialization, considering Eq. (13), stochastic sub-gradient descent is applied to optimize the network efficiently. It should be noted that, the process of calculation is not the same in different layers of the network.

For easy representation, we rewrite Eq. (14) as:

$$J = J_1 + J_2 + J_3, \tag{16}$$

where J_1, J_2 and J_3 indicate the first, second, and third items in Eq. (14), respectively. Let

$$\delta_1^{(m)} = \frac{\partial J_1}{\partial z^{(m)}} \quad \text{and} \quad \delta_2^{(m)} = \frac{\partial J_2}{\partial z^{(m)}}. \tag{17}$$

Accordingly, $\delta_1^{(m)}$ and $\delta_2^{(m)}$ are calculated as

$$\delta_1^{(m)} = \begin{cases} \left(X - A^{(M)}\right) \odot \dot{f}\left(Z^{(M)}\right), & m = M, \\ W^{(m)^T} \cdot \delta_1^{(m+1)} \odot \dot{g}^{(m)}\left(Z^{(m)}\right), & m = 1, \cdots, M-1, \end{cases} \tag{18}$$

and

$$\delta_2^{(m)} = \lambda_1 \cdot \begin{cases} 0, & m = Mid+1, \cdots, M, \\ \left(A^{(\mathrm{Mid})} - A^{(\mathrm{Mid})} \cdot C\right) \odot \dot{g}^{(m)}\left(Z^{(m)}\right) & m = Mid \\ W^{(m)^T} \cdot \delta_1^{(m+1)} \odot \dot{g}^{(m)}\left(Z^{(m)}\right) \quad \text{for} & m = 1, 2, \cdots, Mid-1, \end{cases} \tag{19}$$

respectively, where $\odot$ indicates element-wise multiplication.

Using the chain rule, the update of all parameters $W^{(m)}$ and $b^{(m)}$ are given by

$$\begin{cases} \dfrac{\partial J}{\partial W^{(m)}} & = & \dfrac{\partial J_1}{\partial z^{(m)}} \cdot \dfrac{\partial z^{(m)}}{\partial W^{(m)}} + \dfrac{\partial J_2}{\partial z^{(m)}} \cdot \dfrac{\partial z^{(m)}}{\partial W^{(m)}} + \dfrac{\partial J_3}{\partial W^{(m)}} \\ & = & \left(\delta_1^{(m+1)} + \lambda_1 \delta_2^{(m+1)}\right) \cdot A^{(m)} + \lambda_2 W^{(m)} \\ \dfrac{\partial J}{\partial b^{(m)}} & = & \dfrac{\partial J_1}{\partial z^{(m)}} \cdot \dfrac{\partial z^{(m)}}{\partial b^{(m)}} + \dfrac{\partial J_2}{\partial z^{(m)}} \cdot \dfrac{\partial z^{(m)}}{\partial b^{(m)}} + \dfrac{\partial J_3}{\partial b^{(m)}} \\ & = & \left(\delta_1^{(m+1)} + \lambda_1 \delta_2^{(m+1)}\right) + \lambda_2 b^{(m)} \end{cases} \tag{20}$$

Then,

$$\begin{cases} W^{(m)} \leftarrow W^{(m)} - \alpha \cdot \dfrac{\partial J}{\partial W^{(m)}} \\ b^{(m)} \leftarrow b^{(m)} - \alpha \cdot \dfrac{\partial J}{\partial b^{(m)}}, \end{cases} \tag{21}$$

where α is the learning rate, which is set to 0.001 in the experiment in this paper. To conclude, the LRAE algorithm is described as follows:

4. Experiments and discussion

In this section, we compare the proposed LRAE with other popular subspace clustering methods that have been proposed in recent years on several image datasets.

Table 1
Details of different datasets.

Dataset	Contents	Res.	# Clusters	# Images per class
MNIST	digits	28 × 28	10	1000
COIL100	objects	128 × 128	100	72
ORL	faces	56 × 46	40	10

Table 2
Performance of different subspace clustering algorithms on the MNIST dataset(%).

Alg.	ACC	NMI	ARI
LRAE1	54.4 ± 0.31	49.91 ± 0.65	39.29 ± 0.02
LRAE2	**60.66 ± 1.41**	**61.39 ± 4.10**	**45.7 ± 1.92**
LRAE3	55.49 ± 2.1	61.02 ± 1.00	42.73 ± 0.99
LRR	49.36 ± 1.10	48.88 ± 1.09	41.03 ± 0.92
LRSC	47.7 ± 0.82	43.77 ± 4.35	40.37 ± 1.45
LSR	48.71 ± 0.31	52.71 ± 0.51	40.46 ± 5.73
SSC	36.27 ± 0.32	49.63 ± 0.26	26.80 ± 0.16
AESC	58.31 ± 1.32	61.39 ± 1.89	45.64 ± 2.01
PARTY	58.12 ± 1.12	60.92 ± 0.82	46.69 ± 1.98

The bold values indicate the best performance according to the highest accuracy.

Datasets: The datasets used in our experiments were MNIST [10], COIL-100 [19], and ORL [29]. The MNIST digit database has a training set of 60,000 examples and a testing set of 10,000 examples. It is a widely used benchmark database for testing image analysis algorithms. The size of each image is 28 × 28 and the digits are centered. The samples in this database have a feature dimension of 784. In our experiments, only 10,000 testing examples were used for clustering. The COIL-100 dataset contains 100 objects. Each object is recorded under 72 viewing angles. For convenience, the colorful images are converted to gray-level images; however, this is not necessary. The original images are of size 128 × 128. The ORL dataset consists of 10 frontal face images, each of 40 human subjects. For some subjects, the images are taken at different times and have varied lighting, facial expressions, and facial details. All the images in the ORL database are resized to 56 × 48.

These three datasets contain different types of images, that is, handwritten digits, objects, and facial images. Furthermore, they have different original dimensions. The number of clusters and samples in each class also vary. The comparison of these three datasets is in Table 1.

Experimental settings: We compared the proposed LRAE with well-known subspace algorithms, SSC[6], LRR[16], LRSC[7], LSR[18], AESC[31], and PARTY[24], with different settings.

LRAE: For the proposed LRAE, we used a 350-250-350 network; that is, the activation of neurons in the middle layer were considered as the representation of the original data. Because the dimensions of the original data were different for these three datasets, we applied principal component analysis (PCA) as preprocessing to reduce the original dataset to 350. The two parameters in Eq. (14) were set as follows: $\lambda_1 = 5$ and $\lambda_2 = 3$. When training the neural networks, the learning rate (α) and the number of epochs were set to 1 and 50, respectively. After solving the optimization problem (14), that is, obtaining the representation in the middle layer, three clustering methods, k-means, spectral clustering [21], and SSC, were used to test the performance of our method. Accordingly, the three methods are named LRAE1, LRAE2, and LRAE3, respectively.

AESC: To compare AESC and the LRAE, we implemented AESC with the same network architecture as the LRAE. Moreover, the inputs to AESC were the data after preprocessing by PCA rather than the original data, which was the same for the LRAE.

PARTY: Both PARTY and the LRAE calculated a prior constraint matrix on the middle layer of the network. To compare PARTY and the LRAE, we implemented PARTY with the same network architecture as the LRAE.

LRR methods: For the LRR methods, the parameters were selected to achieve the best performance. The parameter λ was set to 0.1. For LRSC, $\lambda = 20$; λ for LRSC can be set to other values in the range [20,30]. According to the experimental results, this algorithm was not very sensitive to the parameter. For LSR, the parameter was set to 0.02.

SSC: The parameters were set as follows: $\alpha = 20,\ \rho = 1$, and the outlier was considered.

Evaluation: To compare the performance of these algorithms, the clustering accuracy (ACC), normalized mutual information (NMI) and adjusted random index (ARI) were used to evaluate the clustering quality.

4.1. Results

Because the network was initialized randomly, we repeated each experiment in this section more than five times and report the average and standard deviation of three metrics. The clustering results of these algorithms on the three datasets are reported in Tables 2–4.

Table 3
Performance of different subspace clustering algorithms on the COIL-100 dataset(%).

Alg.	ACC	NMI	ARI
LRAE1	51.02 ± 0.04	76.80 ± 0.63	43.61 ± 1.10
LRAE2	54.89 ± 1.73	78.25 ± 1.62	48.99 ± 0.20
LRAE3	**73.03 ± 2.34**	**92.12 ± 0.81**	**65.28 ± 1.03**
LRR	50.01 ± 0.81	46.30 ± 0.77	33.76 ± 0.10
LRSC	40.44 ± 1.03	68.39 ± 0.98	32.82 ± 2.01
LSR	38.21 ± 1.34	65.66 ± 2.03	31.09 ± 1.55
SSC	40.98 ± 0.10	72.18 ± 0.30	38.89 ± 0.10
AESC	53.50 ± 0.20	79.11 ± 0.33	49.23 ± 1.09
PARTY	53.50 ± 0.20	74.11 ± 0.33	42.23 ± 1.09

The bold values indicate the best performance according to the highest accuracy.

Table 4
Performance of different subspace clustering algorithms on the ORL dataset(%).

Alg.	ACC	NMI	ARI
LRAE1	74.62 ± 0.98	88.48 ± 0.57	68.69 ± 1.50
LRAE2	**80.91 ± 1.67**	**90.54 ± 0.72**	**73.53 ± 1.16**
LRAE3	69.86 ± 1.23	85.19 ± 0.84	59.41 ± 2.17
LRR	77.20 ± 0.04	87.53 ± 0.04	69.46 ± 0.06
LRSC	75.42 ± 0.03	84.04 ± 0.05	66.55 ± 0.02
LSR	77.80 ±0.66	88.05 ±0.70	68.36 ±0.86
SSC	76.92 ± 0.85	89.77 ± 0.60	0.70 ± 1.15
AESC	75.30 ± 2.18	89.58 ± 1.15	68.90 ± 3.20
PARTY	79.75 ± 2.61	90.09 ± 1.40	71.61 ± 3.90

The bold values indicate the best performance according to the highest accuracy.

Table 5
Performance(%) of LRAE algorithms with different network architectures.

Architecture	Alg.	ACC	NMI	ARI
350-150-350	LRAE1	74.62 ± 0.98	88.48 ± 0.57	68.69 ± 1.50
	LRAE2	80.91 ± 1.67	90.54 ± 0.72	73.53 ± 2.16
	LRAE3	69.86 ± 1.23	85.19 ± 0.84	59.41 ± 2.17
350-150-500-150-350	LRAE1	64.25 ± 1.12	82.45 ± 2.14	53.17 ± 1.93
	LRAE2	71.50 ± 1.87	85.57 ± 1.33	61.34 ± 2.45
	LRAE3	80.80 ± 1.79	89.98 ± 2.01	71.43 ± 1.73
350-350-350-350-350	LRAE1	63.92 ± 1.67	82.06 ± 1.23	51.73 ± 1.61
	LRAE2	78.75 ± 1.90	88.62 ± 1.91	69.72 ± 1.13
	LRAE3	82.50 ± 1.23	92.43 ± 1.70	77.90 ± 1.80

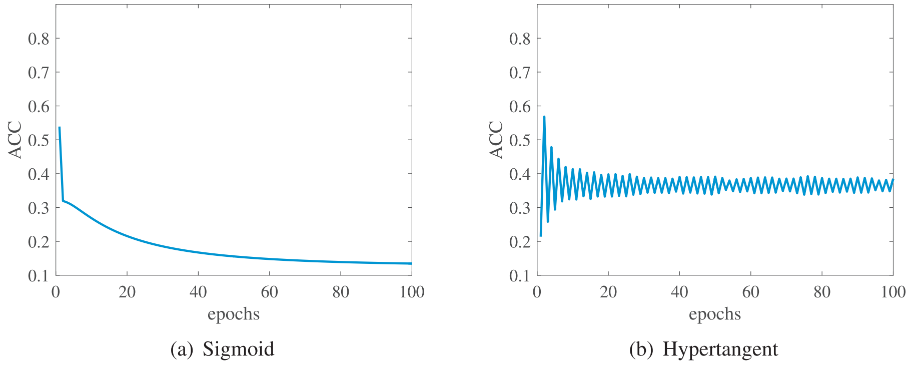
According to the results shown in Tables 2–4, LRAE2 outperformed the other algorithms on the MNIST and ORL datasets. LRAE3 achieved the best results on the COIL-100 dataset. The three LRR based algorithms achieved similar results. Moreover, the LRAE, AESC, and PARTY, which are based on neural networks, performed better than the other algorithms overall.

4.2. Discussion

The LRAE combined an autoencoder with LRR. Therefore, several factors affected the performance, for example, the hyperparameters of the neural network, such as the architecture of the autoencoder, activation function in the network, learning rate and number of epochs, selection of LRR methods, and balance parameters λ_1 and λ_2. In this subsection, we discuss how these factors affected performance.

4.2.1. Architecture of the autoencoder

The experiments in Section 4.1 used a three-layer autoencoder. The different architectures of autoencoders affect their performance. In Table 5, we compare the clustering performance of the three LRAE methods with different networks on the ORL dataset. It can be concluded that, there is no direct relationship between the clustering results and neurons in each layer. Therefore, the architecture of a good network is obtained by experiment.



(a) Sigmoid (b) Hypertangent

Fig. 3. Convergence of the network under different activation functions.

4.2.2. Selection of the activation functions

First, we checked the convergence of the neural network under different activation functions in the hidden layer. In the output layer, we selected a sigmoid function as the activation function because we normalized the input data into [0,1], which has the same range of values as the range of a sigmoid function. As shown in Fig. 3, the network with a sigmoid function in the hidden layer converged in fewer than 50 epochs. By contrast, the LRAE with a hypertangent as the activation function remained oscillating after 100 epochs. Additionally, it can be observed from the figure that the mean squared error (MSE) of the LRAE converged to approximately 0.1 when using sigmoid function, which is greater than the MSE calculated by the autoencoder (approximately 0.05). This is because the LRAE had a low-rank constraint in the middle layer. The cost function contained not only the reconstruction error of each single datum but also the global reconstruction error. Second, we compared the clustering performance of the LRAE methods that selected different activation functions in the hidden layer. From Table 6, it can be observed that the LRAE with a sigmoid function achieved better performance and lower standard deviation than that with a hypertangent function.

4.2.3. Selection of the learning rate and epochs of the network

In addition to the network structure and activation function, the network has other parameters that affect the performance of the LRAE. In this experiment, LRAE2 (obtaining the representation using neural networks, followed by the spectral clustering method) was used because it had the best performance on the ORL dataset, as shown in Table 4. The convergence of the network under different learning rates and number of epochs are shown in Fig. 4. It can be observed that when the learning rate was greater than one, the network converged to relatively small value. Furthermore, we report the clustering accuracy of LRAE2 with different learning rates and numbers of epochs on the ORL dataset. According to Fig. 5, when the learning rate was set to 0.8 and 1, and the number of epochs was greater than 50, the algorithm achieved a relative high performance overall.

4.2.4. Selection of parameters

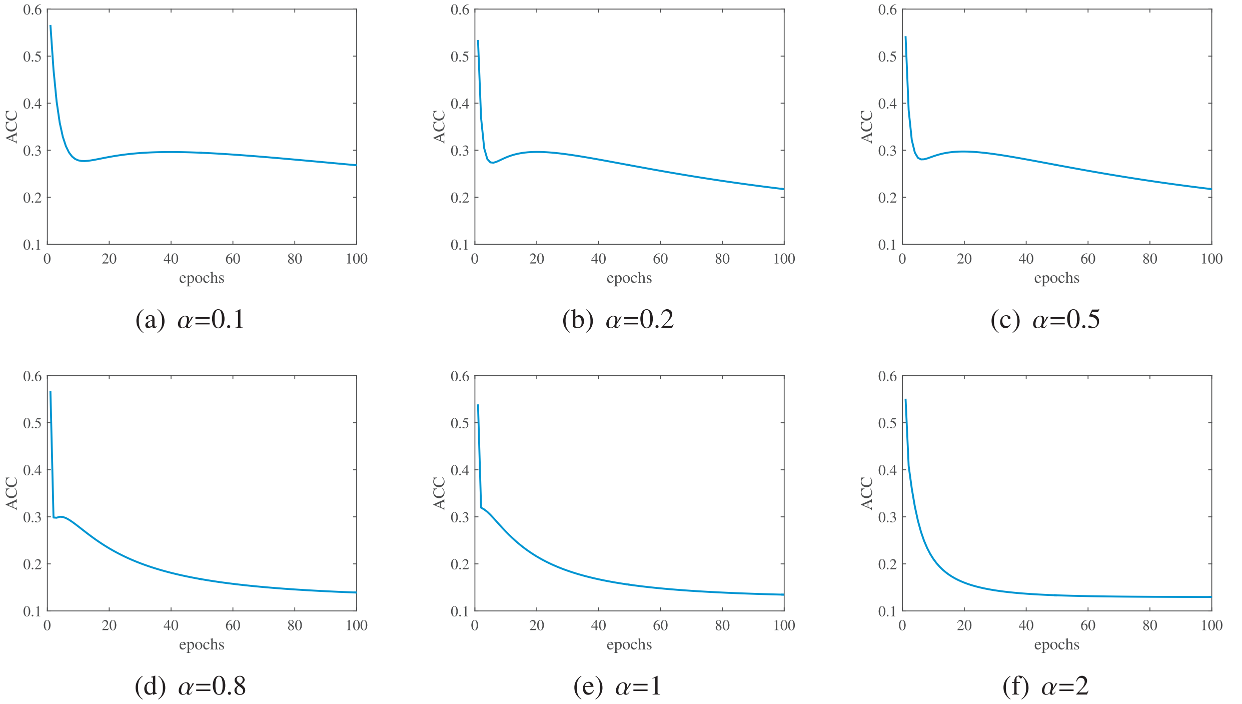
In the proposed algorithm, the two parameters λ_1 and λ_2 are used to balance the two terms in Eq. (14). The selection of these parameters plays an important role for the result. In this experiment, we used the ORL dataset to compare the effect of the parameters. Furthermore, LRAE2 was used because it performed best on the ORL dataset, as shown in Table 4. As shown in Fig. 6, the clustering performance, which is represented by ACC, NMI, and ARI, demonstrated an obvious decline when λ_2 was greater than 30.

4.2.5. Selection of different low-rank algorithms

In the aforementioned experiment for image clustering, the proposed LRAE applied LRR to calculate a low-rank prior constraint matrix. As shown in Table 2–4, there are several algorithms that obtain a prior low-rank matrix: LRR, LRSC, and LSR. We compared the clustering results with different prior low rank constraints. In this experiment, we applied different

Table 6
Performance(%) of LRAE algorithms with different activation functions.

Activation function	Alg.	ACC	NMI	ARI
sigmoid	LRAE1	74.92 ± 0.92	88.58 ± 1.07	68.12 ± 1.22
	LRAE2	81.01 ± 1.27	91.10 ± 0.91	74.34 ± 1.20
	LRAE3	70.83 ± 1.08	85.22 ± 1.04	58.99 ± 1.87
hyper-tangent	LRAE1	73.42 ± 3.7	84.88 ± 2.1	66.7 ± 3.9
	LRAE2	75.98 ± 2.9	86.68 ± 1.8	65.13 ± 4.2
	LRAE3	70.08 ± 2.2	83.35 ± 1.4	61.44 ± 3.3



(a) α=0.1 (b) α=0.2 (c) α=0.5

(d) α=0.8 (e) α=1 (f) α=2

Fig. 4. Convergence of the network under different learning rates and number of epochs.

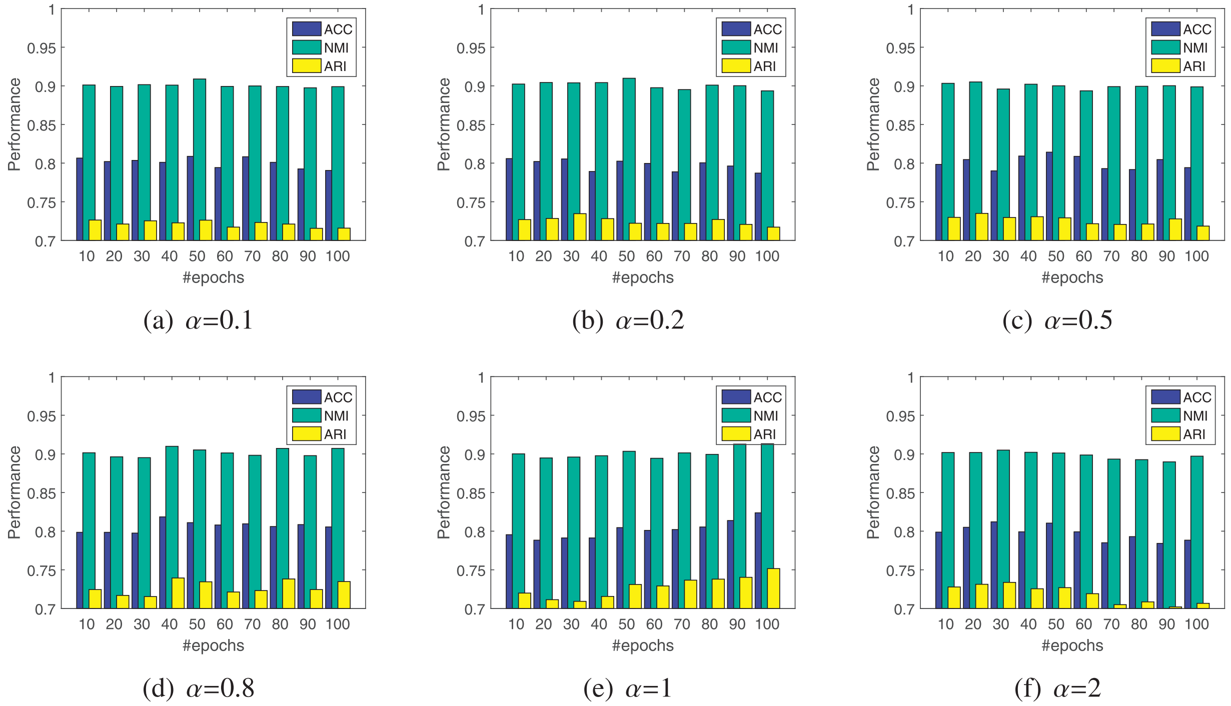
```
Algorithm 1 LRAE.
  Input:
  Input dataset, $X$. Architecture of the network, that is, the number of layers $M$ and the number of neurons in each layer $n^{(m)}$. Initial weights and biases for the network, $W^{(m)}$ and $b^{(m)}$. Appropriate values of parameters, such as $\lambda_1$, $\lambda_2$, and learning rate $\alpha$.
  Prior calculation:
  Calculate the LRR matrix $C$ of $X$ using the algorithm in [16].
  Optimization:
  Perform the feedforward computation of the network using Eq. (12), to obtain the representation.
  while not converge do
      Calculate the gradient using Eq. (20)
      Update $W^{(m)}$ and $b^{(m)}$, for $m = 1, 2, \cdots, M$ using Eq. (21).
  end while
  Obtain the representation in the middle layer by performing the feedforward computation.
  Clustering:
  Perform spectral clustering on the representation $A^{(\mathrm{Mid})}$ via some method.
  Output:
  Clustering results.
```

LRAE methods for different datasets to achieve the best performance; that is, LRAE2 on the MNIST and ORL datasets, and LRAE3 on the COIL-100 dataset. As shown in Table 7, the algorithm with the LRR prior constraint obtained the best results on the MNIST and COIL-100 datasets, whereas the algorithm with the LRSC prior constraint obtained the best results on the ORL dataset.

4.2.6. Selection of different clustering algorithms

In this paper, we focus on the representation of data using the deep neural network method with a low-rank constraint. Actually, after representing data using LRAE, different clustering algorithms can be used for clustering [2,34,36,38]. To compare LARE with other algorithms, three classic clustering methods, that is, k-means, spectral clustering, and SSC, were used to test the performance of LRAE.

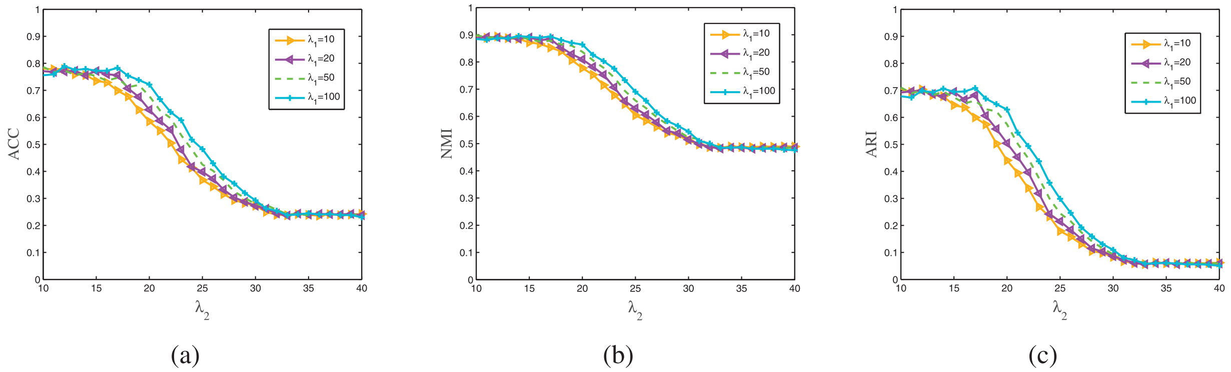
Recently, a number of works have explored combining data clustering with representation learning. For example, in [34], the proposed method learned a mapping from the data space to a lower-dimensional feature space in which it iteratively optimizes a clustering objective. In [36], the author jointly optimized for representation learning and clustering in a re-



(a) α=0.1 (b) α=0.2 (c) α=0.5

(d) α=0.8 (e) α=1 (f) α=2

Fig. 5. Performance of the network under different learning rates and number of epochs.



(a) (b) (c)

Fig. 6. The clustering performance with different parameters for LRAE.

Table 7
Performance(%) of algorithms with different LRR prior constraints.

Dataset	Prior Alg.	ACC	NMI	ARI
MNIST	LRR	60.66 ± 1.41	61.39 ± 2.10	45.70± 1.92
	LRSC	56.10 ± 1.81	60.45 ± 2.02	42.69 ± 1.55
	LSR	54.28 ± 1.65	57.82 ± 2.31	41.91 ± 2.01
COIL-100	LRR	73.03 ± 2.34	92.12 ± 0.81	65.28 ± 1.03
	LRSC	61.34 ± 1.34	81.90 ± 1.63	49.58 ± 1.50
	LSR	55.78 ± 2.23	77.49 ± 2.10	44.56 ± 1.45
ORL	LRR	80.91 ± 1.67	90.54 ± 0.72	73.53 ± 1.16
	LRSC	81.49 ± 2.19	90.77 ± 2.01	73.92 ± 2.11
	LSR	78.19 ± 2.72	87.88 ± 1.68	71.50 ± 1.58

current framework. Compared with classical clustering methods, these deep neural network based algorithms provided an end-to-end learning framework to learn the clusters from an unlabeled dataset. The experimental results demonstrated that these algorithms outperformed most previous clustering methods. In future work, we will consider constructing a unified framework to implement LRR and clustering simultaneously.

5. Conclusion

In this paper, we proposed an LRAE for subspace clustering. We established a novel optimization problem, in which both the self-reconstruction based locality and low-rank based globality of the dataset were mathematically formulated. Extensive experimental results demonstrated that the proposed LRAE outperformed state-of-the-art subspace clustering methods on three types of image datasets. In future work, we will investigate the performance of our method on supervised tasks.

Acknowledgments

This work was supported by the National Science Foundation of China [Grant numbers 61432012, U1435213, and 61332002] and National Key Technology Research and Development Program of the Ministry of Science and Technology of China [Grant number 2014BAH11F01].

References

[1] B.L. Betechuoh, T. Marwala, T. Tettey, Autoencoder networks for hiv classification, Curr. Sci. 91 (11) (2006) 1467–1473.
[2] G. Castellano, A.M. Fanelli, M.A. Torsello, Shape annotation by semi-supervised fuzzy clustering, Inf. Sci. 289 (2014) 148–161.
[3] Y. Chen, L. Zhang, Z. Yi, A novel low rank representation algorithm for subspace clustering, Int. J. Pattern Recognit. Artif. Intell. 30 (04) (2016) 1650007:1–16.
[4] B. Cheng, J. Yang, S. Yan, Y. Fu, T.S. Huang, Learning with l1-graph for image analysis, IEEE Trans. Image Process. 19 (4) (2010) 858–866.
[5] J. Deng, Z. Zhang, F. Eyben, B. Schuller, Autoencoder-based unsupervised domain adaptation for speech emotion recognition, IEEE Signal Process. Lett. 21 (9) (2014) 1068–1072.
[6] E. Elhamifar, R. Vidal, Sparse subspace clustering, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2009, pp. 2790–2797.
[7] P. Favaro, R. Vidal, A. Ravichandran, A closed form solution to robust subspace estimation and clustering, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2011, pp. 1801–1807.
[8] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, J. Mach. Learn. Res. 9 (2010) 249–256.
[9] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.
[10] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, in: Proceedings of the IEEE, 86, 1998, pp. 2278–2324.
[11] G. Liu, P. Li, Low-rank matrix completion in the presence of high coherence, IEEE Trans. Signal Process. 64 (21) (2016) 5623–5633.
[12] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, Y. Ma, Robust recovery of subspace structures by low-rank representation, IEEE Trans. Pattern Anal. Mach. Intell. 35 (1) (2013) 171–184.
[13] G. Liu, Z. Lin, Y. Yu, Robust subspace segmentation by low-rank representation, in: International Conference on Machine Learning, 2010, pp. 663–670.
[14] G. Liu, Q. Liu, P. Li, Blessing of dimensionality: recovering mixture data via dictionary pursuit, IEEE Trans. Pattern Anal. Mach. Intell. 39 (1) (2017) 47–60.
[15] G. Liu, H. Xu, J. Tang, Q. Liu, S. Yan, A deterministic analysis for lrr, IEEE Trans. Pattern Anal. Mach. Intell. 38 (3) (2016) 417–430.
[16] G. Liu, H. Xu, S. Yan, Exact subspace segmentation and outlier detection by low-rank representation, J. Mach. Learn. Res. 22 (2012) 703–711.
[17] G. Liu, S. Yan, Latent low-rank representation for subspace segmentation and feature extraction, in: IEEE International Conference on Computer Vision, 2011, pp. 1615–1622.
[18] C.-Y. Lu, H. Min, Z.-Q. Zhao, L. Zhu, D.-S. Huang, S. Yan, Robust and efficient subspace segmentation via least squares regression, in: European Conference on Computer Vision, Springer, 2012, pp. 347–360.
[19] S.A. Nene, S.K. Nayar, H. Murase, Columbia Object Image Library (COIL-100), Technical Report CUCS-006-96 24, 1996.
[20] A. Ng, Sparse Autoencoder, Technical Report, CS294A Lecture, Standford, 2011.
[21] A.Y. Ng, M.I. Jordan, Y. Weiss, On spectral clustering: analysis and an algorithm, in: Advances in Neural Information Processing Systems, MIT Press, 2001, pp. 849–856.
[22] X. Peng, J. Feng, J. Lu, W.-Y. Yau, Z. Yi, Cascade subspace clustering, in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, 2017, pp. 2478–2484.
[23] X. Peng, J. Lu, Z. Yi, Y. Rui, Automatic subspace learning via principal coefficients embedding, IEEE Trans. Cybern. PP (99) (2016) 1–14, doi:10.1109/TCYB.2016.2572306.
[24] X. Peng, S. Xiao, J. Feng, W.-Y. Yau, Z. Yi, Deep subspace clustering with sparsity prior, in: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, 2016, pp. 1925–1931.
[25] X. Peng, Z. Yu, Z. Yi, H. Tang, Constructing the l2-graph for robust subspace learning and subspace clustering, IEEE Trans. Cybern. 47 (2016) 1053–1066.
[26] X. Peng, L. Zhang, Z. Yi, Scalable sparse subspace clustering, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2013, pp. 430–437.
[27] X. Peng, L. Zhang, Z. Yi, K.K. Tan, Learning locality-constrained collaborative representation for robust face recognition, Pattern Recognit. 47 (9) (2014) 2794–2806.
[28] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning Internal Representations by Error Propagation, Technical Report, DTIC Document, 1985.
[29] F.S. Samaria, A.C. Hartert, Parameterisation of a stochastic model for human face identification, in: Workshop on Applications of Computer Vision, 1994, pp. 138–142.
[30] C.A.P. Sarath, S. Lauly, H. Larochelle, M.M. Khapra, B. Ravindran, V. Raykar, A. Saha, An autoencoder approach to learning bilingual word representations, in: Advances in Neural Information Processing Systems, 2014, pp. 1853–1861.
[31] F. Tian, B. Gao, Q. Cui, E. Chen, T.-Y. Liu, Learning deep representations for graph clustering, in: Proceedings of the 28th AAAI Conference on Artificial Intelligence, 2014, pp. 1293–1299.
[32] R. Vidal, P. Favaro, Low rank subspace clustering (lrsc), Pattern Recognit. Lett. 43 (2014) 47–61.
[33] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion, J. Mach. Learn. Res. 11 (Dec) (2010) 3371–3408.
[34] J. Xie, R. Girshick, A. Farhadi, Unsupervised deep embedding for clustering analysis, in: International Conference on Machine Learning, 2016, pp. 478–487.
[35] J. Xu, L. Xiang, Q. Liu, H. Gilmore, J. Wu, J. Tang, A. Madabhushi, Stacked sparse autoencoder (ssae) for nuclei detection on breast cancer histopathology images, IEEE Trans. Med. Imaging 35 (1) (2016) 119–130.
[36] J. Yang, D. Parikh, D. Batra, Joint unsupervised learning of deep representations and image clusters, in: IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 5147–5156.
[37] Z. Yi, Foundations of implementing the competitive layer model by lotka–volterra recurrent neural networks, IEEE Trans. Neural Networks 21 (3) (2010) 494–507.
[38] L. Zhang, W. Lu, X. Liu, W. Pedrycz, C. Zhong, Fuzzy c-means clustering of incomplete data based on probabilistic information granules of missing values, Knowl. Based Syst. 99 (2016) 51–70.