Structure-Constrained Feature Extraction by Autoencoders for Subspace Clustering

Kewei Tang
Liaoning Normal University
Dalian, Peoples Republic of China
kwtang@lnnu.edu.cn

Kaiqiang Xu
Dalian University of Technology
Dalian, Peoples Republic of China
xukaiqiang@mail.dlut.edu.cn

Zhixun Su
Dalian University of Technology
Dalian, Peoples Republic of China
zxsu@dlut.edu.cn

Wei Jiang
Liaoning Normal University
Dalian, Peoples Republic of China
swxxjw@aliyun.com

Xiaonan Luo
Guilin University of Electronic Technology
Guilin, Peoples Republic of China
luoxn@guet.edu.cn

Xiyan Sun
Guilin University of Electronic Technology
Guilin, Peoples Republic of China
sxy@guet.edu.cn

Abstract

Clustering the data points drawn from a union of subspaces, i.e., subspace clustering, is a hot topic in recent years. The assumption of the nonlinear subspace is more general for the real-world data, but also more difficult for the traditional methods. The deep neural network is a powerful technique extracting nonlinear features, so the autoencoders are usually adopted to handle this unsupervised problem. However, how to add constraints on the features to make them more effective is not heavily addressed by previous work. In this paper, we consider both the global and local structure of the features in the autoencoders by the low-rank property and Laplace operator, respectively. The low-rank property can make the learned features favoring similarity extraction by self-representation and the Laplace operator can help our method explore the useful information in the data set. Note that our way of placing constraints can also be employed in other deep neural networks. In fact, our method is closely associated with previous work. It can be viewed as the more general case of the structured autoencoders. Extensive experiments demonstrate the effectiveness of our method.

1. Introduction

Since many high-dimensional data can be well approximated by a union of low-dimensional subspaces [1, 29, 7, 32], subspace clustering, also called subspace segmentation, has attracted substantial attention in computer vision during the past two decades. Given the data set lies on a union of subspaces, the task of the subspace clustering is to group these data points into different clusters such that each cluster corresponds to a subspace. The review [33] divides the subspace clustering methods into four categories, among which spectral-clustering based methods have been addressed by most recent studies. This kind of methods often learns an affinity matrix encoding the pairwise similarity between data points, and then accomplishes the clustering task by Normalized Cut (NC) [28]. As two popular spectral-clustering based methods, Sparse Subspace Clustering (SSC) [5] and Low-Rank Representation (LRR) [17] obtain the affinity matrix by seeking the sparse and low-rank representation of the data points in the original space, respectively. A lot of extensions have been made about SSC and LRR, such as Least Squares Regression (LSR) [18], Smooth Representation (SMR) [10], and so on. When the mixing subspaces are all linear, these traditional methods can usually achieve promising results.

However, the assumption of linear subspace will be violated in some real-world applications [34]. It is necessary to propose the methods handling the nonlinear subspace well. Unfortunately, some data points can not be expressed as a linear combination of the data points drawn from the same nonlinear subspace, so the self-representation of the data points in the original space cannot be an effective way of building the affinity matrix. In this context, learning features favoring the similarity extraction by self-representation seems more reasonable. As a typical example, Latent Space Sparse Subspace Clustering (LS3C) [21] simultaneously learns the projection of data points and finds the sparse representation in the low-dimensional latent space. Similarly, Subspace Learning based Low-Rank Representation (SLLRR) [30] pursues the low-rank coefficients in the learned subspace. In fact, both LS3C and SLLRR can also be viewed as kernel methods whose idea provides a way of adopting self-representation to deal with the nonlinear case. Since the nonlinear subspace can be transformed into a linear one by some nonlinear mapping, we can learn the affinity matrix by the representation of the transformed data points in the feature space. Hence, some works try to kernelize the traditional subspace clustering methods proposed for the linear subspace to manage the nonlinear case. For example, Kernel Spectral Curvature Clustering (KSCC) [2], Kernel Sparse Subspace Clustering (KSSC) [37, 22] and Robust Kernel Low-Rank Representation (RKLRR) [35] are the kernelizations of Spectral Curvature Clustering [3], SSC and LRR, respectively. The work presenting a Bayesian method named Bayesian Low-rank and Sparse Nonlinear Representation (BLSN) [31] for the nonlinear subspace clustering can also be essentially viewed as a kind of kernel method. However, the performance of these kernel methods heavily relies on the choice of the kernel function, and how to select the appropriate one is still an open problem.

In recent years, deep neural networks have achieved many fruitful results for learning features, so it is a powerful technique to deal with the nonlinear subspace discussed above. However, there are only a few works [24, 23, 11, 39, 38, 12] about subspace clustering because this problem is unsupervised. Since the autoencoders do not require the label information of the data points as the input, they are usually employed to construct novel architecture in this field. Ji et al. [11] add a novel self-expressive layer between the encoder and the decoder in an autoencoder. In this architecture, no restrictions are imposed on the features to make them more effective. Peng et al. propose Structured AutoEncoders (StructAE) [24, 23] to learn the features and then obtain the clustering membership. StructAE constrains the learned features but it only considers their global structure. In fact, the features play a key role in the problem of nonlinear subspace. Moreover, guiding the features favoring affinity extraction seems more reasonable, however, it does not attract enough attentions by previous work. In this paper, we address the problem of how to learn effective features by autoencoders for subspace clustering. Inspired by the traditional subspace clustering methods, we expect the self-representation of the learned features can extract the pairwise similarity well, so the low-rank property is introduced in our model. In addition, the Laplace operator is adopted to keep the local property. Extensive experimental results confirm the effectiveness of our method. In fact, our method is closely related to StructAE. In details, when specifying special parameters, our method will become StructAE. From this point of view, our method can be seen as an extension of StructAE. In summary, our main contributions in this paper lie in the following three aspects:

- We propose a deep learning method for subspace clustering. Different from the idea of previous work, we put emphasis on the constraint of the features. Our method considers both the local and global structure of the features learned by autoencoders via the Laplace operator and low-rank property and achieves state-of-the-art performance.
- The low-rank property of the features will prefer similarity extraction by self-representation. Laplace operator can explore the useful information in the data set. Note that our method provides the way of adding these structure constraint in other deep neural networks.
- Our method can be viewed as the more general case of StructAE.

The remainder of the paper is organized as follows. In Section 2, we introduce the main notations of this paper and make a brief review of StructAE. In Section 3, we propose our method and make a detailed discussion. Comprehensive experimental results are reported in Section 4. We conclude this paper in Section 5.

2. Brief review of StructAE

First, we summarize the main notations used in this paper. We denote vectors by bold lowercase letters, e.g., $\mathbf{x}$, and matrices by bold capital letters, e.g., $\mathbf{X}$. In particular, $\mathbf{I}$ denotes the identity matrix. Note that the bold capital letter with subscript or superscript also denotes the matrix, e.g., $\mathbf{X}_i$. Their entries are denoted by the bold capital letter in the square brackets with subscripts. For instance, $[\mathbf{X}]_{ij}$ denotes the (i, j)-th entry of $\mathbf{X}$; $[\mathbf{X}]_{i:}$ denotes the i-th row of $\mathbf{X}$; $[\mathbf{X}]_{:j}$ is the j-th column of $\mathbf{X}$. For vectors, our notation rules are similar. The trace and the diagonal elements of a square matrix $\mathbf{X}$ are respectively denoted as $tr(\mathbf{X})$ and $diag(\mathbf{X})$. $rank(\mathbf{X})$ represents the rank of the matrix $\mathbf{X}$. $\odot$ denotes the Hadmard product, i.e., $[\mathbf{a} \odot \mathbf{b}]_i = [\mathbf{a}]_i[\mathbf{b}]_i$. $\|\mathbf{X}\|_F = \sqrt{\sum_{ij}[\mathbf{X}]_{ij}^2}$ denotes the Frobenius norm of the matrix $\mathbf{X}$. $\|\mathbf{x}\|_2 = \sqrt{\sum_i[\mathbf{x}]_i^2}$ denotes the ℓ_2-norm of the vector $\mathbf{x}$. $max(\mathbf{x})$ denotes the largest element in $\mathbf{x}$.

In this paper, $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$ always stands for the set of n data points drawn from a union of N subspaces. The discussed neural network always has M layers in total. $\mathbf{h}_i^m \in \mathbb{R}^{d_m}$ and $\mathbf{y}_i^m \in \mathbb{R}^{d_m}$ denote the output and the input of the m-th layer, $m = 1, \cdots, M$, respectively. Particularly, $\mathbf{h}_i^0 = \mathbf{x}_i \in \mathbb{R}^d, i = 1, \cdots, n$ represents the input of the neural network. Denoting f as the activation function, $\mathbf{W}_m \in \mathbb{R}^{d_m \times d_{m-1}}$ as the connection weights from all the units in the $(m-1)$-th layer to those in the m-th layer, $\mathbf{b}_m \in \mathbb{R}^{d_m}$ as the bias associated with the units in the m-th layer, we can get

$$\begin{aligned} &\mathbf{y}_i^m = \mathbf{W}_m \mathbf{h}_i^{m-1} + \mathbf{b}_m, \quad \mathbf{h}_i^m = f(\mathbf{y}_i^m), \\ &i = 1, \cdots, n, \quad m = 1, \cdots, M. \end{aligned} \tag{1}$$

StructAE consists of an encoder and a decoder. In details, after the first $\frac{M}{2}$ hidden layers encoding the data, StructAE learns the features $\mathbf{H}^{\frac{M}{2}} = [\mathbf{h}_1^{\frac{M}{2}}, \cdots, \mathbf{h}_i^{\frac{M}{2}}, \cdots, \mathbf{h}_n^{\frac{M}{2}}]$ which are decoded by the last $\frac{M}{2}$ hidden layers such that the output $\mathbf{H}^M = [\mathbf{h}_1^M, \cdots, \mathbf{h}_i^M, \cdots, \mathbf{h}_n^M]$ approximates the input $\mathbf{X}$ as well as possible. Considering the structure of the learned features and the regularization of the parameters, the objective function of StructAE is formulated in the following

$$\begin{aligned} \min_{\mathbf{W}^m, \mathbf{b}^m} & \tfrac{1}{2}\|\mathbf{X} - \mathbf{H}^M\|_F^2 + \tfrac{\lambda_1}{2}\|\mathbf{H}^{\frac{M}{2}} - \mathbf{H}^{\frac{M}{2}}\mathbf{C}\|_F^2 \\ & + \tfrac{\lambda_2}{2}(\|\mathbf{W}^m\|_F^2 + \|\mathbf{b}^m\|_2^2), \end{aligned} \tag{2}$$

where $\frac{\lambda_1}{2}\|\mathbf{H}^{\frac{M}{2}} - \mathbf{H}^{\frac{M}{2}}\mathbf{C}\|_F^2$ can be viewed as the structured constraint. The input $\mathbf{C}$ can be obtained by the ℓ_1-norm based reconstruction

$$\min_{\mathbf{C}} \|\mathbf{X} - \mathbf{X}\mathbf{C}\|_F^2 + \lambda\|\mathbf{C}\|_1 \quad s.t. \quad diag(\mathbf{C}) = 0 \tag{3}$$

or the ℓ_2-norm based reconstruction

$$\min_{\mathbf{C}} \|\mathbf{X} - \mathbf{X}\mathbf{C}\|_F^2 + \lambda\|\mathbf{C}\|_F \quad s.t. \quad diag(\mathbf{C}) = 0. \tag{4}$$

In their paper, $\mathbf{C}$ learned by the model (3) usually performs better. After learning the features $\mathbf{H}^{\frac{M}{2}}$, StructAE obtains clustering results by applying them to K-means or the spectral clustering-based methods.

3. Structure-Constrained Feature Extraction by Autoencoders

The structured constraint in StructAE only considers the globality. In fact, it is reasonable to place constraints on features to make them more effective. Many interesting properties deserve the attempt in the autoencoders. We will consider both the local and global structure in our method as follows.

3.1. Our Method

Since we aim to extract the features such that their self-representation can effectively encode the pairwise similarity, inspired by the idea in traditional methods, we suppose these features are globally low-rank. Hence, the cost function of our autoencoders is first considered as

$$\min_{\mathbf{W}^m, \mathbf{b}^m} \frac{1}{2}\|\mathbf{X} - \mathbf{H}^M\|_F^2 + rank(\mathbf{H}^{\frac{M}{2}}). \tag{5}$$

However, $rank(\mathbf{H}^{\frac{M}{2}})$ is non-smooth. In addition, as the commonly used convex surrogate, the nuclear norm is also non-smooth. According to the reference [25], the mission of minimizing the rank of $\mathbf{H}^{\frac{M}{2}}$ can also be accomplished by the following model

$$\min_{\mathbf{A}, \mathbf{B}} \|\mathbf{A}\|_F^2 + \|\mathbf{B}\|_F^2 \quad s.t. \quad \mathbf{H}^{\frac{M}{2}} = \mathbf{A}\mathbf{B}. \tag{6}$$

Therefore, our cost function is revised as

$$\begin{aligned} \min_{\mathbf{W}^m, \mathbf{b}^m, \mathbf{A}, \mathbf{B}} & \tfrac{1}{2}\|\mathbf{X} - \mathbf{H}^M\|_F^2 + \gamma(\|\mathbf{A}\|_F^2 + \|\mathbf{B}\|_F^2) \\ & + \alpha\|\mathbf{H}^{\frac{M}{2}} - \mathbf{A}\mathbf{B}\|_F^2. \end{aligned} \tag{7}$$

However, the above function only considers the global structure. To capture the local capture, we add another term about $\mathbf{H}^{\frac{M}{2}}$ in the cost function

$$\begin{aligned} \min_{\mathbf{W}^m, \mathbf{b}^m, \mathbf{A}, \mathbf{B}} & \tfrac{1}{2}\|\mathbf{X} - \mathbf{H}^M\|_F^2 + \gamma(\|\mathbf{A}\|_F^2 + \|\mathbf{B}\|_F^2) \\ & + \alpha\|\mathbf{H}^{\frac{M}{2}} - \mathbf{A}\mathbf{B}\|_F^2 + \beta\Sigma_{ij}[\mathbf{S}]_{ij}\|\mathbf{h}_i^{\frac{M}{2}} - \mathbf{h}_j^{\frac{M}{2}}\|_2^2, \end{aligned} \tag{8}$$

where $\mathbf{S}$ is a symmetric matrix. The large $[\mathbf{S}]_{ij}$ will make $\mathbf{h}_i^{\frac{M}{2}}$ close to $\mathbf{h}_j^{\frac{M}{2}}$, so the local information of the feature $\mathbf{h}_i^{\frac{M}{2}}$ is recorded in $[\mathbf{S}]_{i:}$ or $[\mathbf{S}]_{:i}$. Similar terms have been constructed in many previous work to keep the local structure [8, 12]. Compared to the recent work [12], we make a detailed discussion about how to construct the appropriate weight matrix $\mathbf{S}$ in Section 3.3. It is well-known that

$$\Sigma_{ij}[\mathbf{S}]_{ij}\|\mathbf{h}_i^{\frac{M}{2}} - \mathbf{h}_j^{\frac{M}{2}}\|_2^2 = tr(\mathbf{H}^{\frac{M}{2}}\mathbf{L}(\mathbf{H}^{\frac{M}{2}})^T), \tag{9}$$

where $\mathbf{L}$ is the Laplace matrix of the weight matrix $\mathbf{S}$. Hence, our cost function can be formulated as

$$\begin{aligned} \min_{\mathbf{W}^m, \mathbf{b}^m, \mathbf{A}, \mathbf{B}} & \tfrac{1}{2}\|\mathbf{X} - \mathbf{H}^M\|_F^2 + \gamma(\|\mathbf{A}\|_F^2 + \|\mathbf{B}\|_F^2) \\ & + \alpha\|\mathbf{H}^{\frac{M}{2}} - \mathbf{A}\mathbf{B}\|_F^2 + \beta tr(\mathbf{H}^{\frac{M}{2}}\mathbf{L}(\mathbf{H}^{\frac{M}{2}})^T). \end{aligned} \tag{10}$$

Actually, our method is closely associated with StructAE. By means of $\|\mathbf{E}\|_F^2 = tr(\mathbf{E}\mathbf{E}^T)$, we can obtain

$$\|\mathbf{H}^{\frac{M}{2}} - \mathbf{H}^{\frac{M}{2}}\mathbf{C}\|_F^2 = tr(\mathbf{H}^{\frac{M}{2}}(\mathbf{I} - \mathbf{C})(\mathbf{I} - \mathbf{C})^T(\mathbf{H}^{\frac{M}{2}})^T) \tag{11}$$

This conclusion implies that StructAE can be viewed as a special case of our method by setting $\mathbf{L} = (\mathbf{I} - \mathbf{C})(\mathbf{I} - \mathbf{C})^T$, $\gamma = 0$, and $\alpha = 0$.

3.2. The Algorithm

In this section, we will provide the algorithm of our method. Inspired by the idea of Alternating Direction Method (ADM) [16], we explore a scheme optimizing each variable alternatively, until the function value in Eq.(10) converges or a maximum number of iterations is reached. When $\mathbf{W}^m, \mathbf{b}^m, m = 1, \cdots, M$ and $\mathbf{B}$ are fixed, we minimize $\mathbf{A}$ in Eq.(10) as follows

$$\min_{\mathbf{A}} \gamma\|\mathbf{A}\|_F^2 + \alpha\|\mathbf{H}^{\frac{M}{2}} - \mathbf{AB}\|_F^2. \tag{12}$$

The closed form solution is

$$\mathbf{A} = (\alpha\mathbf{H}^{\frac{M}{2}}\mathbf{B}^T)(\alpha\mathbf{B}\mathbf{B}^T + \gamma\mathbf{I})^{-1}. \tag{13}$$

Similarly, fixing $\mathbf{W}^m, \mathbf{b}^m, m = 1, \cdots, M$ and $\mathbf{A}$, we can obtain the closed form solution of $\mathbf{B}$

$$\mathbf{B} = (\alpha\mathbf{A}^T\mathbf{A} + \gamma\mathbf{I})^{-1}(\alpha\mathbf{A}^T\mathbf{H}^{\frac{M}{2}}). \tag{14}$$

In the last, we will show our strategy for $\mathbf{W}^m, \mathbf{b}^m, m = 1, \cdots, M$ with $\mathbf{A}$ and $\mathbf{B}$ fixed. In this situation, the function in Eq.(10) becomes

$$\begin{aligned}\min_{\mathbf{W}^m,\mathbf{b}^m} & \tfrac{1}{2}\|\mathbf{X} - \mathbf{H}^M\|_F^2 + \alpha\|\mathbf{H}^{\frac{M}{2}} - \mathbf{AB}\|_F^2 \\ & + \beta tr(\mathbf{H}^{\frac{M}{2}}\mathbf{L}(\mathbf{H}^{\frac{M}{2}})^T).\end{aligned} \tag{15}$$

In order to efficiently solve it by the commonly used s-tochastic gradient descent algorithm, we rewrite it in the sample-wise form

$$\begin{aligned} g = & \textstyle\sum_i(\frac{1}{2}\|\mathbf{x}_i - \mathbf{h}_i^M\|_2^2 + \alpha\|\mathbf{h}_i^{\frac{M}{2}} - [\mathbf{AB}]_{:i}\|_2^2 \\ & + \beta\Sigma_j[\mathbf{S}]_{ij}\|\mathbf{h}_i^{\frac{M}{2}} - \mathbf{h}_j^{\frac{M}{2}}\|_2^2).\end{aligned} \tag{16}$$

By means of a back-propagation algorithm, we can compute the gradients of $\mathbf{W}^m, \mathbf{b}^m, m = 1, \cdots, M$ and update them in the following.

$$\mathbf{W}^m = \mathbf{W}^m - \mu\frac{\partial g}{\partial \mathbf{W}^m}, \quad \mathbf{b}^m = \mathbf{b}^m - \mu\frac{\partial g}{\partial \mathbf{b}^m}. \tag{17}$$

We skip the details for the derivation of these gradients and provide the results as follows

$$\frac{\partial g}{\partial \mathbf{W}^m} = (\mathbf{\Delta}^m + 2\alpha\mathbf{\Lambda}^m + 2\beta\mathbf{\Theta}^m)(\mathbf{h}_i^{(m-1)})^T, \tag{18}$$

$$\frac{\partial g}{\partial \mathbf{b}^m} = \mathbf{\Delta}^m + 2\alpha\mathbf{\Lambda}^m + 2\beta\mathbf{\Theta}^m, \tag{19}$$

where $\mathbf{\Delta}^m$ is

$$\begin{cases} -(\mathbf{x}_i - \mathbf{h}_i^M) \odot f'(\mathbf{y}_i^m) & m = M, \\ (\mathbf{W}^{m+1})^T\mathbf{\Delta}^{m+1} \odot f'(\mathbf{y}_i^m) & otherwise. \end{cases} \tag{20}$$

$\mathbf{\Lambda}^m$ is

$$\begin{cases} (\mathbf{h}_i^{\frac{M}{2}} - [\mathbf{AB}]_{:i}) \odot f'(\mathbf{y}_i^m) & m = \frac{M}{2}, \\ (\mathbf{W}^{m+1})^T\mathbf{\Lambda}^{m+1} \odot f'(\mathbf{y}_i^m) & m = 1, \cdots, \frac{M}{2} - 1, \\ 0 & m = \frac{M}{2} + 1, \cdots, M. \end{cases} \tag{21}$$

$\mathbf{\Theta}^m$ is

$$\begin{cases} \mathbf{H}^{\frac{M}{2}}[\mathbf{L}]_{i:}^T \odot f'(\mathbf{y}_i^m) & m = \frac{M}{2}, \\ (\mathbf{W}^{m+1})^T\mathbf{\Theta}^{m+1} \odot f'(\mathbf{y}_i^m) & m = 1, \cdots, \frac{M}{2} - 1, \\ 0 & m = \frac{M}{2} + 1, \cdots, M. \end{cases} \tag{22}$$

The detailed procedure is provided in Algorithm 1.

Algorithm 1 The algorithm of the proposed method

```
Input: The data set $\mathbf{X}$, the weight matrix $\mathbf{S}$ obtained from
  Algorithm 2 and the parameters $\alpha$, $\beta$, $\gamma$.
Initialize: $\mathbf{W}^m$, $\mathbf{b}^m$, $m = 1, \cdots, M$.
  for $m = 1 : M$ do
    Compute $\mathbf{H}^m = [\mathbf{h}_1^m, \cdots, \mathbf{h}_i^m, \cdots, \mathbf{h}_n^m]$ by Eq. (1).
  end for
  while not converged do
    for $i = 1 : n$ do
      Randomly select a data point $\mathbf{x}_i$.
      for $m = 1 : M$ do
        Update $\mathbf{h}_i^m$ by Eq. (1).
      end for
      for $m = M : 1$ do
        Update $\mathbf{W}^m$, $\mathbf{b}^m$ by Eq. (17)-(22).
      end for
      Update $\mathbf{A}$ by Eq. (13).
      Update $\mathbf{B}$ by Eq. (14).
    end for
  end while
  Obtain the clustering results by $\mathbf{H}^{\frac{M}{2}}$.
```

3.3. The Weight Matrix

In this subsection, we make a detailed discussion about the weight matrix $\mathbf{S}$ in our method. In order to keep the local structure, we construct

$$\mathbf{S} = \mathbf{R} \odot \mathbf{T}, \tag{23}$$

where $\mathbf{R}$ and $\mathbf{T}$ include the information from the neighbors and local linear subspace, respectively. In details,

$$\mathbf{R} = \begin{cases} 1 & if \quad \mathbf{x}_i \in knn(\mathbf{x}_j) \quad or \quad \mathbf{x}_j \in knn(\mathbf{x}_i), \\ 0 & otherwise, \end{cases} \tag{24}$$

where $knn(\mathbf{x}_j)$ denotes the k nearest neighbors of $\mathbf{x}_j$. In fact, if the data points far from the intersection of the subspaces are close to each other, they are usually in the same subspace, so we build $\mathbf{R}$ to collect the local information of this type. However, $\mathbf{R}$ cannot handle the intersection of the subspaces well. According to previous work [26, 27], the global nonlinear subspaces can locally be well-approximated by a series of local linear subspaces. Hence, we construct $\mathbf{T}$ to overcome the limitation of $\mathbf{R}$ as follows

$$\mathbf{T} = \begin{cases} 1 & if \quad \mathbf{x}_i \in N(\mathbf{x}_j) \quad and \quad \mathbf{x}_j \in N(\mathbf{x}_i), \\ 0 & otherwise, \end{cases} \tag{25}$$

where $N(\mathbf{x}_j)$ denotes the set of data points in the local linear subspace of $\mathbf{x}_j$. The selection of $N(\mathbf{x}_j)$ shall give priority to the data points almost linearly correlated with $\mathbf{x}_j$. Therefore, we consider the metric based on the angle in the following

$$\mathbf{F} = \widetilde{\mathbf{G}}^T \widetilde{\mathbf{G}}, \tag{26}$$

$$[\mathbf{G}]_{ij} = exp(-\frac{1-(\widetilde{\mathbf{x}}_i^T \widetilde{\mathbf{x}}_j)^2}{\tau}), \quad i,j = 1, \cdots, n \tag{27}$$

where τ is the mean of $1-(\widetilde{\mathbf{x}}_i^T \widetilde{\mathbf{x}}_j)^2, i,j = 1,\cdots,n$, $\widetilde{\mathbf{x}}_i$ is the normalization of $\mathbf{x}_i$, i.e., $\|\widetilde{\mathbf{x}}_i\|_2 = 1$, and similarly $[\widetilde{\mathbf{G}}]_{:i}$ is the normalization of $[\mathbf{G}]_{:i}$, $i = 1, \cdots, n$. Under this definition, when the angle between $\mathbf{x}_i$ and $\mathbf{x}_j$ is larger, $[\mathbf{G}]_{ij}$ will be larger. With respect to the data points linearly correlated, the corresponding columns of $\mathbf{G}$ will have large elements in similar locations, so i-th row of $\mathbf{F}$ measures the linear correlation degree between all the data points and $\mathbf{x}_i$ by considering the whole data set. Hence, $N(\mathbf{x}_j)$ are selected from the first p data points except $\mathbf{x}_j$ itself sorted by $[\mathbf{F}]_{j:}$ in the descending order. The detailed procedure is provided in Algorithm 2.

More discussions about the idea of the main steps in the Algorithm 2 are in the following. Since the local linear subspace shall include some linearly correlated data points, its rank, relative to the number of its data points, may usually be not high. Denoted as ξ and recorded in ϕ in Algorithm 2, the rank is one of the main measures to evaluate which data set is the local linear subspace in our algorithm. The computation of this quantity mainly depends on the evaluations on the singular values of the current data set $[\mathbf{x}_j, \mathbf{x}_{j_1}, \cdots, \mathbf{x}_{j_i}]$ in step 1. Our idea is utilizing the percentage and the relative variation recorded in $\mathbf{u}$ and $\mathbf{v}$, respectively. Given a data set, if the sum of r singular values has a high percentage of the sum of all the singular values and the relative variation of the r-th singular value to the $(r+1)$-th singular value is also large, the rank of this data set may be r. Hence, we find the maximal component of φ, equally considering $\mathbf{u}$ and $\mathbf{v}$, to determine the rank of the data set. To some extent, $max(\varphi)$ can be viewed as the confidence of the obtained rank. However, if all the singular values do not have a dramatic change, the data set is possibly full rank. Hence, we add a judgment in our algorithm. We expect the local linear subspace is low-rank but includes the data points as many as possible, so we first construct ψ to consider the rank ξ, the number of the data points including $\mathbf{x}_j$ and the confidence of the rank $max(\varphi)$, and then compute $max(\psi)$ to determine the key parameters of the local linear subspace. The meaning of λ and $[\phi]_\lambda$ are the number of the data points and the rank, respectively. Since we suppose the rank of the local linear subspace is relatively small to the number of its data points, $\frac{\lambda}{[\phi]_\lambda}$ can be viewed as the confidence of $N(\mathbf{x}_j)$. Hence, the local linear subspace only includes the first q data points. When $\mathbf{x}_j$ is linearly correlated with $\mathbf{x}_i$, $\mathbf{x}_i$ will also be linearly correlated with $\mathbf{x}_j$. Therefore, we use "and" in the construction of $\mathbf{T}$ in Eq. (25).

Algorithm 2 The algorithm about $N(\mathbf{x}_j)$

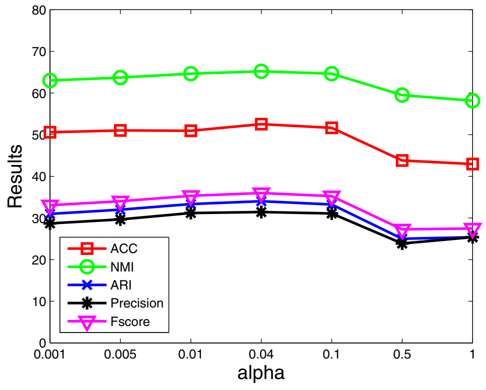
```
Input: The first $p$ data points $\mathbf{x}_{j_1}, \cdots, \mathbf{x}_{j_p}$ except $\mathbf{x}_j$ sorted
  by $[\mathbf{F}]_{j:}$ in Eq. (26) in the descending order, and $\varepsilon$.
Output: $N(\mathbf{x}_j) = \{\mathbf{x}_{j_1}, \cdots, \mathbf{x}_{j_q}\}$.
  for $i = 1 : p$ do
    1. Compute all the singular values $\sigma_1 \geq \cdots \geq \sigma_{i+1}$ of
    the data matrix $[\mathbf{x}_j, \mathbf{x}_{j_1}, \cdots, \mathbf{x}_{j_i}]$.
    2. $s = \Sigma_{k=1}^{i} \sigma_k$.
    for $l = 1 : i$ do
      1. $t = \Sigma_{k=1}^{l} \sigma_k$.
      2. $[\mathbf{u}]_l = \frac{t}{s}$.
      3. $[\mathbf{v}]_l = \frac{\sigma_l - \sigma_{l+1}}{\sigma_{l+1}}$.
    end for
    if $max(\mathbf{v}) < \varepsilon$ then
      1. $[\phi]_i = i + 1$,
      2. $[\psi]_i = 1 + [\mathbf{v}]_i$.
    else
      1. $\varphi = \mathbf{u} + \frac{\mathbf{v}}{max(\mathbf{v})}$.
      2. Compute $max(\varphi)$ and its index $\xi$.
      3. $[\phi]_i = \xi$.
      4. $[\psi]_i = \frac{max(\varphi) * (i+1)}{\xi}$.
    end if
  end for
  1. Compute $max(\psi)$ and its index $\lambda$.
  2. Compute the nearest integer of $\frac{\lambda}{[\phi]_\lambda}$ and denote it as $q$.
```

4. Experimental Results

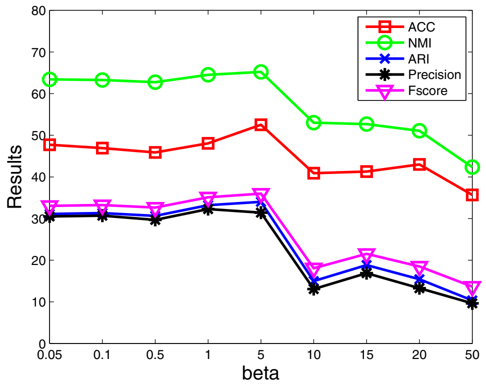
In this section, we test the performance of our method on four benchmark databases. Since StructAE can be viewed as a special case of our method to some extent, we compare it in our experiment. In addition, some popular subspace clustering methods for linear or nonlinear subspace are also compared in the experiment, including LRR, SSC, LSR, SMR, KLRR, KSSC. Note that LSR1 and LSR2 denote two versions of LSR, please refer to the work [18] for details. With regard to the kernel methods, how to choose an appropriate kernel is still an open problem. Hence, we report their results by selecting the common kernel. KLRRG and KLRRP denote KLRR with Gaussian kernel and Polynomi-

Method	ACC	NMI	ARI	Precision	Fscore
Ours	52.53±0.23	65.21±0.30	34.02±0.27	31.41±0.30	36.00±0.25
StructAE	45.65±1.06	50.24±1.45	20.84±0.28	22.25±0.28	26.61±0.23
LRR	38.69±1.31	49.93±0.51	14.38±3.24	17.73±3.50	21.84±3.03
SSC	37.71±0.61	32.46±0.81	9.90±0.35	8.89±0.27	13.45±0.32
LSR1	41.50±1.22	47.00±0.54	11.00±1.30	17.39±1.31	17.65±1.18
LSR2	41.40±0.89	47.76±0.78	11.93±1.67	18.32±1.76	18.10±1.53
SMR	44.77±0.46	48.12±1.00	18.55±2.62	14.04±3.42	19.11±2.46
KSSCG	41.93±1.11	37.50±0.42	13.09±0.69	11.16±0.64	16.45±0.60
KSSCP	42.39±1.17	38.08±1.10	13.85±0.85	11.72±0.71	15.23±0.77
KLRRG	36.30±0.58	46.49±0.53	13.70±0.90	10.97±1.12	15.93±0.81
KLRRP	19.87±0.52	35.73±0.54	9.99±0.28	12.06±0.27	12.39±0.27

Table 1. The results of all the compared methods on LPQ of Extended Yale B.



(a)



(b)

Figure 1. (a) The results of our methods versus α on LPQ of Extended Yale B. (b) The results of our methods versus β on LPQ of Extended Yale B

al kernel, respectively. Similarly, KSSCG and KSSCP have the same meaning. Since accuracy (ACC) and normalized mutual information (NMI) are two key evaluation metrics in subspace clustering, our experiment compares the above methods on these two measures. Moreover, the adjusted rand index (ARI), Precision, and Fscore are also reported to make a comprehensive evaluation. With respect to these five evaluation metrics, larger value always means better performance. Because NC includes the step of K-means, we repeat the experiment five times and report the mean and standard deviation to reduce the impact of randomization. The important parameters of all the compared methods are empirically tuned on the source codes provided by the original authors to achieve the best result.

4.1. Implementation Details of Our Method

The implementation details of our method are similar with StructAE for a fair comparison. In summary, the activation function is

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{28}$$

Hence,

$$f'(x) = 1 - (f(x))^2. \tag{29}$$

In the experiment, our method includes five layers consisting of $300 - 200 - 150 - 200 - 300$ neurons. The initialization is completed by training shallow networks of $300-200-300$ and $200-150-200$, respectively. The idea is inspired by the strategy in [9]. In our implementation, the fully connected layers are adopted. However, other architectures such as convolutional neural networks can also be used. When the loss is not larger than 10^{-3}, our method is believed to achieve convergence. In the training process, the batch-size is 1. The iteration of the Algorithm 1 is stopped if our method converges or the training epoch reaches to 100. Similar with StructAE, our method also obtains clustering results by applying the learned features $\mathbf{H}^{\frac{M}{2}}$ to K-means or the spectral clustering-based methods.

4.2. The Results on LPQ of Extended Yale B

The Extended Yale B [6] database consists of 2414 facial images captured from 38 persons, where the size of each image is 192×168. Following [23], we first divide each image

Method	ACC	NMI	ARI	Precision	Fscore
Ours	85.75±0.40	90.50±0.26	80.46±0.39	79.82±0.36	81.45±0.37
StructAE	84.38±1.67	89.72±0.57	81.51±2.02	81.65±2.83	82.43±1.90
LRR	75.99±4.01	88.64±1.37	76.12±3.03	69.67±3.92	74.58±2.83
SSC	75.53±1.82	88.01±0.85	74.53±1.91	70.86±1.63	75.88±1.82
LSR1	69.57±1.54	77.81±0.88	61.53±1.38	60.66±1.22	63.51±1.31
LSR2	69.82±2.66	77.10±1.24	61.20±2.33	60.72±2.94	63.20±2.19
SMR	77.88±1.17	86.80±0.12	73.63±1.38	71.99±2.30	74.98±1.29
KSSCG	71.89±1.54	84.23±1.19	66.84±2.51	63.64±3.55	68.60±2.34
KSSCP	71.39±0.51	84.08±0.36	67.85±0.72	63.76±1.80	69.59±0.66
KLRRG	68.58±0.91	79.64±0.60	61.44±1.21	58.33±1.12	63.51±1.14
KLRRP	64.28±1.18	76.55±0.63	56.37±1.61	54.36±2.40	58.69±1.48

Table 2. The results of all the compared methods on LPQ of Coil 20.

Method	ACC	NMI	ARI	Precision	Fscore
Ours	22.72±0.58	10.02±0.25	5.22±0.25	13.26±0.19	18.68±0.18
StructAE	21.05±0.56	7.54±0.78	3.57±0.71	12.23±0.46	16.72±0.57
LRR	15.06±0.09	2.53±0.02	0.73±0.01	10.54±0.01	12.09±0.02
SSC	16.02±0.03	3.44±0.03	1.47±0.02	11.27±0.01	11.54±0.01
LSR1	16.67±0.03	3.88±0.00	1.75±0.00	11.52±0.00	11.76±0.00
LSR2	16.65±0.00	3.79±0.00	1.69±0.00	11.47±0.00	11.70±0.00
SMR	17.30±0.00	3.98±0.00	1.71±0.00	11.28±0.00	13.09±0.00
KSSCG	20.47±0.04	6.78±0.00	4.48±0.00	11.85±0.00	14.36±0.00
KSSCP	18.92±0.11	5.72±0.09	2.64±0.05	11.34±0.04	12.42±0.05
KLRRG	20.35±0.00	6.79±0.00	4.52±0.00	12.02±0.00	14.11±0.00
KLRRP	19.81±0.19	6.73±0.04	3.15±0.03	11.76±0.03	13.01±0.03

Table 3. The results of all the compared methods on LPQ of CIFAR 10.

into multiple non-overlapping 15×15 blocks and then extract the local phase quantization (LPQ) feature [20]. In the last, we project these features into 300-dimensional space by PCA for computational efficiency. Since StructAE performs better on this database when employing the Homotopy algorithm [36] to solve the model (3), we also obtain our clustering results by applying the learned features $\mathbf{H}^{\frac{M}{2}}$ to the model (3) for a fair comparison in this section. The experimental results are reported in Table 1 where our method performs best on all the measures. In particular, our NMI is much better than that of the others. In addition, we evaluate the impact of the main parameters on our method. We vary α and β by fixing the other parameters and plot the results in Fig. 1 (a) and (b), respectively. Since our method performs best at $\alpha = 0.04$ and $\beta = 5$, the range of α and β are set as $[0.001, 1]$ and $[0.05, 50]$, respectively. Although our performance has a little fluctuation, our results are usually better than the compared methods. These results imply the effectiveness of our method.

4.3. The Results on LPQ of Coil 20

The Coil 20 [19] database contains 1440 grayscale images from 20 objects. With respect to each object, 72 images are acquired by a fixed camera. The dimension of these images is 32×32. In our experiment, LPQ features are extracted from 8×8 non-overlapping patch and then projected into a 300-dimensional space by PCA. In this section, the strategy of obtaining the clustering results is the same as that of Extended Yale B. Table 2 shows the experimental results of all the compared methods. Although ARI, Precision and Fscore of our method are only a few lower than those of StructAE, our method performs best on two main measures, i.e., ACC and NMI. This result also indicates the effectiveness of our method.

4.4. The Results on CIFAR 10

The CIFAR 10 [13] database includes 60000 color images in 10 classes, with 6000 images per class. The size of these images is also 32×32. In our experiment, we select the first 200 images for each class and convert them into gray images. In this section, we compare the methods on both the histogram of oriented gradients (HOG) [4] and LPQ feature by following the setting of [23]. Since StructAE performs better when applying learned features $\mathbf{H}^{\frac{M}{2}}$ to K-means on this database, our method adopts the same strategy for a fair comparison. Table 3 and 4 report the re-

Method	ACC	NMI	ARI	Precision	Fscore
Ours	27.00±0.28	13.16±0.12	6.86±0.10	15.72±0.13	16.98±0.06
StructAE	25.92±0.95	12.85±0.53	6.62±0.37	15.88±0.35	16.02±0.31
LRR	15.25±0.11	2.06±0.03	0.69±0.02	10.58±0.02	10.86±0.02
SSC	18.40±0.07	4.66±0.03	2.01±0.02	11.66±0.01	12.45±0.07
LSR1	15.61±0.33	2.50±0.12	0.89±0.06	10.77±0.06	10.84±0.05
LSR2	15.52±0.49	2.43±0.14	0.85±0.10	10.74±0.09	10.81±0.09
SMR	20.92±0.04	6.18±0.02	3.02±0.02	12.54±0.02	13.23±0.01
KSSCG	23.60±0.00	11.22±0.01	6.04±0.01	14.01±0.01	14.57±0.01
KSSCP	23.70±0.00	11.86±0.00	6.47±0.00	14.26±0.00	15.00±0.00
KLRRG	23.35±0.00	11.90±0.00	6.70±0.00	14.09±0.00	15.14±0.00
KLRRP	23.30±0.00	12.00±0.00	6.66±0.00	14.24±0.00	14.32±0.00

Table 4. The results of all the compared methods on HOG of CIFAR10.

Method	ACC	NMI	ARI	Precision	Fscore
Ours	63.16±0.11	50.96±0.04	38.69±0.12	44.87±0.09	45.27±0.10
StructAE	59.77±0.00	48.28±0.05	33.99±0.11	40.60±0.09	41.08±0.10
LRR	52.15±0.00	43.83±0.00	28.37±0.00	33.59±0.00	36.67±0.00
SSC	42.19±3.13	34.01±3.05	15.53±1.59	19.33±1.01	28.99±1.15
LSR1	52.15±0.00	45.94±0.00	27.02±0.00	29.61±0.00	36.59±0.00
LSR2	54.10±0.00	48.26±0.00	30.36±0.00	33.01±0.00	39.12±0.00
SMR	48.83±0.00	41.03±0.00	25.67±0.00	28.72±0.00	35.39±0.00
KSSCG	49.22±0.00	43.05±0.00	26.90±0.00	31.41±0.00	35.73±0.00
KSSCP	50.98±0.00	44.95±0.00	28.07±0.00	31.53±0.00	37.03±0.00
KLRRG	51.37±0.00	45.11±0.00	22.40±0.00	25.61±0.00	33.02±0.00
KLRRP	48.67±0.35	40.28±0.34	18.28±0.26	22.57±0.18	29.70±0.23

Table 5. The results of all the compared methods on Caltech 101.

sults of all the methods on LPQ and HOG, respectively. Our method also performs best on these two features even if we adopt another clustering strategy. This result confirms the effectiveness of our method.

4.5. The Results on Caltech 101

Caltech 101 [15] database consists of color images with different sizes from 101 objects. We select images from ten objects, named camera, pagoda, dollarbill, panda, pizza, wrench, starfish, yin-yang, windsor-chair, inline-skate, and convert them into gray images to conduct our experiment. We first extract the descriptors of the spatial pyramid bag of words [14] with the size of the dictionary set at 100, and then project these descriptors into the 300-dimensional space by PCA. In the last, the normalized data are used to compare all the methods. In this section, we try another popular spectral clustering-based method to make a comprehensive comparison. In details, our method obtains the clustering results by applying the learned features to LRR. For a fair comparison, we also adopt the same strategy for StructAE. The experimental results are shown in Table 5 where our method still achieves the best results on all the measures. This result further demonstrates the effectiveness of our method.

5. Conclusion and Future Work

In this paper, we study the strategy of adding the structure constraint on the features to guide the autoencoders. Both the local and global structure are considered by the Laplace operator and low-rank property, respectively. The effectiveness of our method is confirmed by the extensive experiments. In addition, our way of placing structure constraint can also be adopted by other deep neural networks. With respect to the future work, since the extensions of some traditional subspace clustering methods can also be applied in the graph-based semi-supervised learning, we will try our method on this problem in the future, too. One of our advantages may be that the weight matrix in our method can include more accurate information provided by the labeled data.

Acknowledgments This work is partially supported by the National Natural Science Foundation of China (Nos. 61702243, 61572099, 61976041, 61771229, 61320106008, 61772149, and 61561016), and the National Science and Technology Major Project (No. 2018ZX04041001-007).

References

[1] R. Basri and D. W. Jacobs. Lambertian reflectance and linear subspaces. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(2):218–233, 2003.

[2] G. Chen, S. Atev, and G. Lerman. Kernel spectral curvature clustering (KSCC). In *ICCV*, pages 765–772, 2009.

[3] G. Chen and G. Lerman. Spectral curvature clustering (SC-C). *International Journal of Computer Vision*, 81(3):317–330, 2009.

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, pages 886–893, 2005.

[5] E. Elhamifar and R. Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(11):2765–2781, 2013.

[6] A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(6):643–660, 2001.

[7] T. Hastie and P. Y. Simard. Metrics and models for handwritten character recognition. *Statistical Science*, 13(1):54–65, 1998.

[8] X. He, S. Yan, Y. Hu, P. Niyogi, and H. Zhang. Face recognition using laplacianfaces. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(3):328–340, 2005.

[9] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[10] H. Hu, Z. Lin, J. Feng, and J. Zhou. Smooth representation clustering. In *CVPR*, pages 3834–3841.

[11] P. Ji, T. Zhang, H. Li, M. Salzmann, and I. D. Reid. Deep subspace clustering networks. In *NIPS*, pages 23–32, 2017.

[12] Q. Ji, Y. Sun, J. Gao, Y. Hu, and B. Yin. Nonlinear subspace clustering via adaptive graph regularized autoencoder. *IEEE Access*, 7:74122–74133, 2019.

[13] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Technical Report*, 2009.

[14] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, pages 2169–2178, 2006.

[15] F. Li, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1):59–70, 2007.

[16] Z. Lin, M. Chen, and Y. Ma. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *CoRR*, abs/1009.5055, 2010.

[17] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma. Robust recovery of subspace structures by low-rank representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(1):171–184, 2013.

[18] C.-Y. Lu, H. Min, Z.-Q. Zhao, L. Zhu, D.-S. Huang, and S. Yan. Robust and efficient subspace segmentation via least squares regression. In *ECCV*, pages 347–360, 2012.

[19] S. A. Nene, S. K. Nayar, and H. Murase. Columbia object image library (coil-20). *Technical Report, CUCS-005-96*, 1996.

[20] V. Ojansivu and J. Heikkilä. Blur insensitive texture classification using local phase quantization. In *ICISP*, pages 236–243, 2008.

[21] V. M. Patel, H. V. Nguyen, and R. Vidal. Latent space sparse subspace clustering. In *ICCV*, pages 225–232, 2013.

[22] V. M. Patel and R. Vidal. Kernel sparse subspace clustering. In *ICIP*, pages 2849–2853, 2014.

[23] X. Peng, J. Feng, S. Xiao, W. Yau, J. T. Zhou, and S. Yang. Structured autoencoders for subspace clustering. *IEEE Trans. Image Processing*, 27(10):5076–5086, 2018.

[24] X. Peng, S. Xiao, J. Feng, W. Yau, and Z. Yi. Deep subspace clustering with sparsity prior. In *IJCAI*, pages 1925–1931, 2016.

[25] B. Recht, M. Fazel, and P. A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501, 2010.

[26] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[27] L. K. Saul and S. T. Roweis. Think globally, fit locally: Unsupervised learning of low dimensional manifold. *Journal of Machine Learning Research*, 4:119–155, 2003.

[28] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, 2000.

[29] P. Y. Simard, Y. LeCun, and J. S. Denker. Efficient pattern recognition using a new transformation distance. In *NIPS*, pages 50–58, 1992.

[30] K. Tang, X. Liu, Z. Su, W. Jiang, and J. Dong. Subspace learning based low-rank representation. In *ACCV*, pages 416–431, 2016.

[31] K. Tang, J. Zhang, Z. Su, and J. Dong. Bayesian low-rank and sparse nonlinear representation for manifold clustering. *Neural Processing Letters*, 44(3):719–733, 2016.

[32] R. Tron and R. Vidal. A benchmark for the comparison of 3-d motion segmentation algorithms. In *CVPR*, 2007.

[33] R. Vidal. Subspace clustering. *IEEE Signal Process. Mag.*, 28(2):52–68, 2011.

[34] Y. Wang, Y. Jiang, Y. Wu, and Z. Zhou. Spectral clustering on multiple manifolds. *IEEE Trans. Neural Networks*, 22(7):1149–1161, 2011.

[35] S. Xiao, M. Tan, D. Xu, and Z. Y. Dong. Robust kernel low-rank representation. *IEEE Trans. Neural Netw. Learning Syst.*, 27(11):2268–2281, 2016.

[36] A. Y. Yang, S. S. Sastry, A. Ganesh, and Y. Ma. Fast *l*1-minimization algorithms and an application in robust face recognition: A review. In *ICIP*, pages 1849–1852, 2010.

[37] M. Yin, Y. Guo, J. Gao, Z. He, and S. Xie. Kernel sparse subspace clustering on symmetric positive definite manifolds. In *CVPR*, pages 5157–5164, 2016.

[38] T. Zhang, P. Ji, M. Harandi, W. Huang, and H. Li. Neural collaborative subspace clustering. In *ICML*, pages 7384–7393, 2019.

[39] P. Zhou, Y. Hou, and J. Feng. Deep adversarial subspace clustering. In *CVPR*, pages 1596–1604, 2018.