Received 7 July 2023, accepted 17 July 2023, date of publication 25 July 2023, date of current version 1 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3298693

RESEARCH ARTICLE

Sparse Subspace Learning Based on Learnable Constraints for Image Clustering

SIYUAN ZHAO
International School, Beijing University of Posts and Telecommunications, Beijing 100080, China
e-mail: zsy2210820800@163.com

ABSTRACT Sparse subspace clustering is a widely used method for clustering high dimensional data, but the traditional method is complex and requires prior information that can be difficult to obtain in unsupervised scenarios. In this paper, we propose a new method called Self-constrained Sparse Subspace Clustering (ScSSC) that adds two self-constraints to find prior information, simplifying the clustering of high dimensional data. The proposed algorithm is a non-deep neural network model that extends the traditional sparse subspace clustering objective function and transforms the clustering problem into a spectral clustering optimization problem. The algorithm can discover a high-quality cluster structure without prior information, making it highly effective in unsupervised scenarios. Our experiment analysis shows that the proposed algorithm outperforms other comparison methods in terms of three metrics. The algorithm's robustness and stability are further demonstrated through ablation experiments and parameter analysis. The proposed algorithm reduces the complexity of the clustering method, making it a valuable tool in understanding and analyzing information in datasets.

INDEX TERMS Subspace clustering, sparse subspace clustering, self-constrained clustering.

I. INTRODUCTION

Clustering is a common data analysis technique that partitions objects in a dataset into different groups or categories, such that objects within the same group exhibit high similarity, while those in different groups have low similarity [1], [2], [3]. Subspace clustering, a special clustering method, has been widely studied and applied in many areas, such as computer vision, bio-informatics, and social network analysis. The basic principle of subspace clustering is to represent each object in the dataset as a vector and decompose these vectors into several subspace, followed by clustering in each subspace. The advantages of subspace clustering are its ability to handle high-dimensional data and to discover multiple subspace within a dataset, thereby better capturing the inherent structure of the data. Moreover, subspace clustering can handle noise and outliers in the dataset, thereby improving the accuracy and robustness of clustering. In summary, subspace clustering is a powerful clustering method that can help us better understand and analyze information in datasets [4], [5], [6].

Sparse subspace clustering is a technique for clustering data points that lie in a union of low-dimensional subspaces. The method assumes that data points within the same subspace can be represented as linear combinations of a few basis vectors. Sparse subspace clustering applies self-representation to learn a sparse graph, which can be used to cluster the data points. The method was first proposed in [7] and has since been extended and improved by other researchers. Some related techniques include robust subspace clustering, which uses low-rank representation to segment subspace [8], and deep subspace clustering, which applies deep learning techniques to learn a hierarchical representation of the data [9]. Other methods use multi-kernel techniques or constrained Laplacian rank algorithms to learn a robust graph [10], [11], [12].

The optimization problem for traditional sparse subspace clustering involves finding a sparse representation of the data points in terms of a dictionary of atoms that span the subspace. This is typically formulated as an ℓ_1-norm minimization problem, where the objective function is the sum

The associate editor coordinating the review of this manuscript and approving it for publication was Mounim A. El Yacoubi.

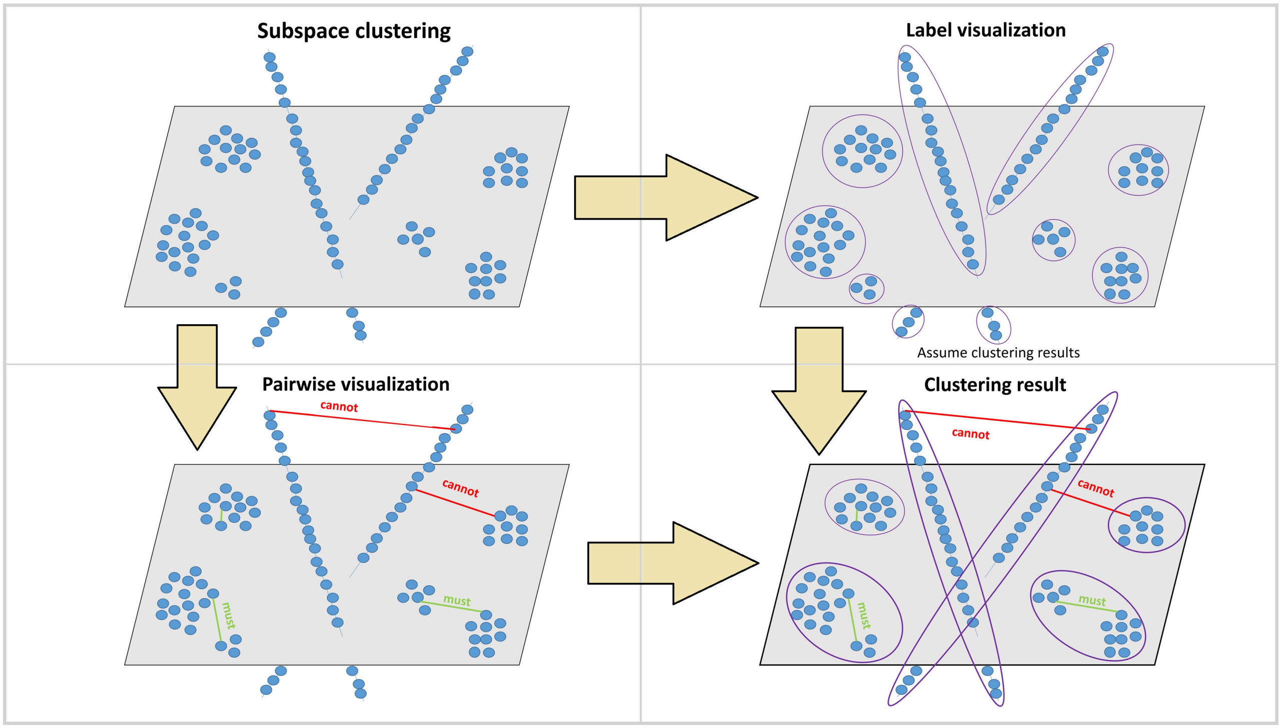


FIGURE 1. **Illustration of the proposed method.**

of the ℓ_1-norm of the sparse representations subject to a constraint that enforces consistency with the subspace model. The steps of finding the feature matrix are very complex (such as alternating direction method of multipliers (ADMM) or proximal gradient descent), and many functions are used in the process, resulting in relatively complex time complexity [7]. The optimization problem for traditional spectral clustering is relatively tractable, which involves finding the eigenvectors of the Laplacian matrix of the data graph and using them to embed the data points into a low-dimensional space. This embedding is then clustered using a standard clustering algorithm such as k-means [13], [14], [15]. Based on the above analysis, an intuitive improvement to sparse subspace clustering lies in the optimization method, that is, the sparse subspace clustering problem is transformed into spectral clustering problem, and the corresponding optimization method can be greatly simplified.

The use of prior information can enhance the performance of an algorithm on downstream tasks, which is a commonsense in the field of machine learning. Although the sparse subspace clustering algorithm has theoretical advantages, it also needs prior information to improve the performance of the algorithm in terms of clustering accuracy and other metrics, and the improvement of performance is often positively correlated with the degree of prior information. On one hand, it is difficult to gain prior information in unsupervised scenarios. On the other hand, inaccurate prior information can not improve the clustering results, but reduce the clustering effect [16]. In recent years, how to improve the performance of algorithms in unsupervised scenarios has become a hot research topic. The key is to find a substitute for prior information, that is, to improve the performance of algorithms without prior information. Fortunately, in recent years, the idea and means of learning self-constrained information from existing data to guide the learning process have been proposed, which provides a new direction for the performance improvement of unsupervised sparse subspace clustering [17], [18], [19].

In this paper, in order to extract effective subspace structure under the unsupervised scene and in a more straightforward manner, we propose a novel sparse subspace clustering model, called self-constrained sparse subspace clustering (ScSSC) algorithm. Fig.1 gives the illustration of our proposed method. Specifically, the main contributions of this paper are as follows.

- The proposed method extends the objective function of traditional sparse subspace clustering by adding pairwise and label self-constrained terms to it, which can help to improve the clustering performance without prior information and reduce the difference between the clustering results and users' expectations.
- Inspired by the optimization of spectral clustering optimization problem, which involves finding the eigenvectors of the Laplacian matrix of the data graph, we simplify the optimization process of sparse subspace clustering into a more straightforward process.
- Extensive experiment analysis shows that the proposed algorithm outperforms other comparison methods in terms of three metrics, and the algorithm's robustness

and stability are further demonstrated through ablation experiments and parameter analysis.

The outline of the rest of this paper is described as follows. Section II describes the related work of this paper. In Section III, We present the theory and implementation of our self-constrained subspace clustering algorithm in detail. In Section IV, we demonstrate the proposed method's performance and effectiveness through experiments. Section V summarizes the article and makes a future outlook for the proposed method.

II. RELATED WORK

A. UNSUPERVISED SUBSPACE CLUSTERING

Unsupervised subspace clustering is a popular research topic in the field of pattern analysis and machine intelligence. One approach to unsupervised subspace clustering is based on sparse representation, where each data point is represented as a linear combination of a few basis vectors from a dictionary [7]. Another approach is based on low-rank representation, where each data point is represented as a linear combination of other data points in the same subspace [10]. Other methods include spectral clustering with constraints [15], subspace clustering via joint L2,1-norm low-rank and sparse representation [6], and robust principal component analysis-based subspace clustering [20]. The limits of unsupervised subspace clustering are that it assumes that the data lies in a union of low-dimensional subspace, which may not be true for all datasets, and it requires the number of subspace to be known or estimated beforehand, which can be difficult in practice. Additionally, unsupervised subspace clustering may not perform well when the subspace are highly overlapping or when there is noise in the data.

B. SEMI-SUPERVISED CLUSTERING

Semi-supervised clustering is a popular research topic in the field of pattern analysis and machine intelligence. One approach to semi-supervised clustering is based on graph-based methods, where a graph is constructed based on pairwise similarity between data points, and the labels of the labeled data points are propagated to the unlabeled data points via graph diffusion [21]. Another approach is based on subspace clustering with constraints, where the subspace structure of the data is used as a constraint to guide the clustering process [22]. Other methods include constrained spectral clustering [23], co-training-based clustering [24], and self-training-based clustering [25]. Semi-supervised clustering can effectively reduce the difference between the clustering results and users' expectations by leveraging the availability of labeled data to guide the clustering process. However, semi-supervised clustering also has its limitations. One limitation is that it requires a small amount of labeled data, which may not always be available or may be expensive to obtain.

C. SELF-CONSTRAINED CLUSTERING

Self-supervised clustering with deep models has recently gained attention as a promising approach to unsupervised learning. One popular method is based on contrastive learning, where the model learns to distinguish between similar and dissimilar pairs of data points [26]. Another method is based on clustering-based objectives, where the model learns to cluster the data points in an unsupervised manner [27]. A third method is based on generative modeling, where the model learns to generate realistic samples from the data distribution [28]. Other methods include self-supervised representation learning via pretext tasks [29] and self-supervised learning via multi-task objectives [30]. While self-supervised clustering with deep models has shown promising results, one limitation is that they often require large amounts of data and computational resources to train effectively. Another limitation is that the quality of the learned representations can be highly dependent on the choice of hyperparameters and architecture, which can be difficult to optimize in practice [31].

III. THE PROPOSED METHOD

A. PRELIMINARY

Sparse subspace clustering (SSC) is a popular technique for clustering high-dimensional data that lie in low-dimensional subspace. The basic idea behind SSC is to represent each data point as a linear combination of other data points, with the constraint that only a few coefficients are nonzero. Mathematically, given an input data matrix $\mathbf{X}$ of size $n \times d$, we seek to find a sparse representation matrix $\mathbf{Z}$ of size $n \times n$ such that:

$$\min_{\mathbf{Z}} ||\mathbf{X} - \mathbf{Z}\mathbf{X}||_F^2 + \lambda||\mathbf{Z}||_1 \tag{1}$$

where λ is a regularization parameter that controls sparsity.

Spectral clustering is a graph-based clustering method that aims to partition data points into groups based on their similarity. The Laplacian matrix $\mathbf{L}$ of the graph is decomposed into its eigenvectors and eigenvalues, which are used to embed the data points into a low-dimensional space. The objective function of spectral clustering is typically formulated as a trace minimization problem:

$$\min_{\mathbf{Y}} \mathrm{Tr}(\mathbf{Y}^T\mathbf{L}\mathbf{Y}) \quad \text{s.t.} \quad \mathbf{Y}^T\mathbf{Y} = \mathbf{I} \tag{2}$$

where $\mathbf{Y}$ is an $n \times k$ indicator matrix that encodes the cluster assignments. Once the data points are embedded in this space, they can be clustered using standard techniques such as k-means.

In spectral clustering, we construct a similarity matrix $\mathbf{S}$ using pairwise or other types of similarities. Let $\mathbf{S}_{ij}$ be the similarity between data points $\mathbf{x}_i$ and $\mathbf{x}_j$, then we have:

$$\mathbf{S}_{ij} = \begin{cases} \exp\left(-\dfrac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right) & \text{if } \mathbf{x}_j \in \mathrm{kNN}(\mathbf{x}_i), \\ 0, & \textit{otherwise} \end{cases} \tag{3}$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are the feature vectors of data points $\mathbf{x}_i$ and $\mathbf{x}_j$, respectively, σ is a parameter that controls the width of the Gaussian kernel, and $\mathrm{kNN}(\mathbf{x}_i)$ denotes the set of indices of the k-nearest neighbors of data point $\mathbf{x}_i$. The value of k determines how many nearest neighbors are considered when constructing the similarity matrix. If $\mathbf{x}_j$ is not one of the k-nearest neighbors of $\mathbf{x}_i$, then $S_{ij} = 0$. The diagonal elements are set to 0 to ensure that each data point is not similar to itself. The similarity matrix $\mathbf{S}$ is sparse because each data point is only similar to a few other data points.

According to our observation, we find that the inverse problem of sparse subspace is spectral decomposition problem based on sparse similarity matrix. Specifically, for SSC, we first construct sparse similarity matrix $\mathbf{Z}$ about $\mathbf{X}$, then the inverse problem of SSC is as follows.

$$\min_{\mathbf{Y}} ||\mathbf{Y} - \mathbf{ZY}||_F^2 \tag{4}$$

A further transformation is as follows.

$$\begin{aligned} ||\mathbf{Y} - \mathbf{ZY}||_F^2 &= ||(\mathbf{I} - \mathbf{Z})\mathbf{Y}||_F^2 \\ &= \mathrm{Tr}(\mathbf{Y}^T(\mathbf{I} - \mathbf{Z})^T(\mathbf{I} - \mathbf{Z})\mathbf{Y}) = \mathrm{Tr}(\mathbf{Y}^T\mathbf{L}_Z\mathbf{Y}) \end{aligned} \tag{5}$$

where $\mathbf{L}_Z = \mathbf{I} - (\mathbf{Z} + \mathbf{Z}^T - 2\mathbf{Z}^T\mathbf{Z})$. So we can solve an inverse problem to obtain sparse representations using eigenvalue decomposition efficiently.

B. SELF-CONSTRAINED SPARSE SUBSPACE CLUSTERING

The pairwise self-constrained term is designed to enhance the similarity matrix of spectral clustering. It does this by penalizing pairs of data points that are assigned to different clusters but have a high similarity score. Mathematically, the pairwise self-constrained term can be expressed as $\Omega_P = ||\mathbf{YY}^T - \mathbf{P}||_F^2$, where $\mathbf{P}$ is a $n \times n$ pairwise variable matrix to save pairwise self-supervised constraints.

The label self-constrained term improves the label features of spectral clustering by encouraging data points with similar labels to be assigned to the same cluster. This helps to ensure that data points with similar labels are more likely to be assigned to the same cluster, which can also improve the accuracy of the clustering results. Mathematically, the label self-constrained term can be expressed as $\Omega_B = ||\mathbf{Y} - \mathbf{B}||_F^2$, where $\mathbf{B}$ is a $n \times k$ label variable matrix to save label self-supervised constraints.

By simultaneously using these two self-constrained terms in SSC, we can take advantage of their complementary strengths and improve both the similarity matrix and label features of spectral clustering. This can lead to more accurate and robust clustering results, especially when prior information about the data is limited or unavailable.

Combining two self-constrained terms with the objective function of sparse subspace clustering, the optimization problem become

$$\begin{aligned} &\min_{\mathbf{P},\mathbf{B},\mathbf{Y}} \Omega = ||\mathbf{Y} - \mathbf{ZY}||_F^2 + \alpha||\mathbf{YY}^T - \mathbf{P}||_F^2 + \beta||\mathbf{Y} - \mathbf{B}||_F^2 \\ &\text{s.t.}, \mathbf{Y}^T\mathbf{Y} = \mathbf{I}_k, ||\mathbf{P}_{i.}||_2^2 = 1, ||\mathbf{B}_{i.}||_2^2 = 1 \end{aligned} \tag{6}$$

where $\mathbf{Z}$ is the predefined sparse similarity matrix about $\mathbf{X}$, α is used to control the contribution of pairwise self-constrained term and β is the weight of label self-constraint item. Unlike the semi-supervised clustering algorithm, $\mathbf{P}$ and $\mathbf{B}$ are treated as variables in the new optimization problem. At the same time, the roles of $||\mathbf{P}_{i.}||_2^2 = 1$ and $||\mathbf{B}_{i.}||_2^2 = 1$ are to use the normalization of L_2 norm to make $\mathbf{P}$ and $\mathbf{B}$ sparse, which strengthens the high reliable values and weakens the low values.

C. OPTIMIZATION ALGORITHM

To solve the problem at hand, we can apply a well-known technique called alternating optimization. This approach involves fixing a set of variables and optimizing the objective function with respect to the remaining variables by alternating between them. In our case, we will start by fixing the matrices $\mathbf{Y}$ and $\mathbf{P}$ and then update the matrix $\mathbf{B}$ in one step. Next, we will fix $\mathbf{Y}$ and $\mathbf{B}$ and update $\mathbf{P}$ in another step. Finally, we will fix the matrices $\mathbf{P}$ and $\mathbf{B}$ and update $\mathbf{Y}$ in the last step. By iteratively repeating these steps until convergence, we can efficiently arrive at a solution for our problem.

Update P while fixing B and Y: When the matrix $\mathbf{Y}$ is fixed, the objective function can be resolved into two independent subproblems. The first subproblem is to minimize Ω_1 as follows:

$$\min_{\mathbf{P}} \Omega_1 = ||\mathbf{YY}^T - \mathbf{P}||_F^2, s.t., ||\mathbf{P}_{i\cdot}||_2^2 = 1 \tag{7}$$

and the second subproblem is to minimize Ω_2 as follows:

$$\min_{\mathbf{B}} \Omega_2 = ||\mathbf{Y} - \mathbf{B}||_F^2, s.t., ||\mathbf{B}_{i\cdot}||_2^2 = 1. \tag{8}$$

To find the optimal solution for $\mathbf{P}$, we need to compute the gradient of the sub-objective function and set it equal to zero. This gives us the following equations:

$$\Delta\Omega_1 = \sum_{i=1}^{n}\sum_{j=1}^{n}[(\mathbf{Y}_{i.}\mathbf{Y}_{j.}^T - \mathbf{P}_{ij})^2 + \eta(\mathbf{P}_{ij}^2 - 1)] \tag{9}$$

Solving these equations gives us the optimal solution for $\mathbf{P}$, which can be expressed as

$$\begin{cases} \dfrac{\partial\Delta\Omega_1}{\partial\mathbf{P}_{ij}} = 2\mathbf{P}_{ij} - 2\mathbf{Y}_{i.}\mathbf{Y}_{j.}^T + 2\eta\mathbf{P}_{ij} = 0 \\ \displaystyle\sum_{j=1}^{n}\mathbf{P}_{ij}^2 = 1. \end{cases} \tag{10}$$

Therefore, we can obtain the optimal solution for $\mathbf{P}$ as follows:

$$\tilde{\mathbf{P}}_{ij} = \sqrt{\frac{(\mathbf{YY}^T)_{ij}^2}{\sum_{r=1}^{n}(\mathbf{YY}^T)_{ir}^2}} \tag{11}$$

If we calculate the optimal solution for $\mathbf{P}$ directly with the formula, they may lose their final state values. To avoid this, we take a gradual approach to obtain the final state, we combine the previous state of $\mathbf{P}$ with the optimal solution:

$$\mathbf{P}^{(t)} \leftarrow \mathbf{P}^{(t-1)} + \lambda\tilde{\mathbf{P}}. \tag{12}$$

where $\lambda \in [0, 1]$ is a weight of the new states.

Update P while fixing B and Y: Similarly, according to the method of updating Y and using the Lagrange multiplier method, it can be transformed into:

$$\Delta\Omega_2 = \sum_{i=1}^{n}\sum_{j=1}^{k}[(\mathbf{Y}_{ij} - \mathbf{B}_{ij})^2 + \eta(\mathbf{B}_{ij}^2 - 1)] \tag{13}$$

According to the constraint conditions and gradient 0, the optimal solution of **B** satisfies:

$$\begin{cases} \dfrac{\partial\Delta\Omega_2}{\partial\mathbf{B}_{ij}} = 2\mathbf{B}_{ij} - 2\mathbf{Y}_{ij} + 2\eta\mathbf{B}_{ij} = 0 \\ \displaystyle\sum_{j=1}^{k}\mathbf{B}_{ij}^2 = 1. \end{cases} \tag{14}$$

The optimal solution of **B** is:

$$\tilde{\mathbf{B}}_{ij} = \sqrt{\frac{\mathbf{Y}_{ij}^2}{\sum_{r=1}^{k}\mathbf{Y}_{ir}^2}} \tag{15}$$

Similarly, we combine the state at the previous time of **B** with the optimal solution:

$$\mathbf{B}^{(t)} \leftarrow \mathbf{B}^{(t-1)} + \lambda\tilde{\mathbf{B}}. \tag{16}$$

$\lambda \in [0, 1]$ is the same weight of the new states as for the optimization of **P**. In this article, we set the λ value to decrease as the number of iterations increases. The reason is that the weight of the self-supervised information extracted from **Y** is decreasing in the iterative process. In other words, the last sentence means to lean step by step towards the optimal solution. Therefore, we can use negative exponents to set λ

$$\lambda^{(t)} = \exp(-t) \tag{17}$$

Once we have solved for **P** and **B**, we can move on to the next step.

Update Y while fixing B and P: When **P** and **B** are fixed, since

$$\begin{aligned} ||\mathbf{Y}\mathbf{Y}^T - \mathbf{P}||_F^2 &= \mathrm{Tr}((\mathbf{Y}\mathbf{Y}^T - \mathbf{P})^T(\mathbf{Y}\mathbf{Y}^T - \mathbf{P})) \\ &= 2\mathrm{Tr}(\mathbf{I} - \mathbf{Y}^T\mathbf{P}\mathbf{Y}) \end{aligned} \tag{18}$$

we can get the following simplified optimization subproblem:

$$\begin{aligned} \min_{\mathbf{Y}} \Omega &= ||\mathbf{Y} - \mathbf{Z}\mathbf{Y}||_F^2 + \alpha||\mathbf{Y}\mathbf{Y}^T - \mathbf{P}||_F^2 + \beta||\mathbf{Y} - \mathbf{B}||_F^2 \\ &= \mathrm{Tr}(\mathbf{Y}^T(\mathbf{I} - (\mathbf{Z} + \mathbf{Z}^T - 2\mathbf{Z}^T\mathbf{Z}))\mathbf{Y}) \\ &\quad + 2\alpha\mathrm{Tr}(\mathbf{I} - \mathbf{Y}^T\mathbf{P}\mathbf{Y}) + \beta||\mathbf{Y} - \mathbf{B}||_F^2 \\ &= \mathrm{Tr}(\mathbf{Y}^T\hat{\mathbf{L}}_Z\mathbf{Y}) + \beta||\mathbf{Y} - \mathbf{B}||_F^2 \end{aligned} \tag{19}$$

where $\hat{\mathbf{L}}_Z = \mathbf{I} - (\mathbf{Z} + \mathbf{Z}^T - 2\mathbf{Z}^T\mathbf{Z} + 2\alpha\mathbf{P}) = \mathbf{I} - (\hat{\mathbf{Z}} + 2\alpha\mathbf{P})$, and **I** is the identity matrix of appropriate size.

Then the problem becomes a label propagation problem. In this case, its closed-form solution $\hat{\mathbf{Y}}$ is:

$$\hat{\mathbf{Y}} = \frac{\beta}{1+\beta}(\mathbf{I} - \frac{1}{1+\beta}(\hat{\mathbf{Z}} + 2\alpha\mathbf{P}))^{-1}\mathbf{B} \tag{20}$$

Usually, people don't calculate $(\mathbf{I} - \frac{1}{1+\beta}(\hat{\mathbf{Z}} + 2\alpha\mathbf{P}))^{-1}$ directly. Due to

$$\begin{aligned} &(\mathbf{I} - \frac{1}{1+\beta}(\hat{\mathbf{Z}} + 2\alpha\mathbf{P}))^{-1} \\ &\quad = \lim_{t\to\infty}\Bigg[(\frac{1}{1+\beta}(\hat{\mathbf{Z}} + 2\alpha\mathbf{P}))^{t-1} \\ &\quad + \frac{\beta}{1+\beta}\sum_{i=0}^{t-1}(\frac{1}{1+\beta}(\hat{\mathbf{Z}} + 2\alpha\mathbf{P}))^{i}\Bigg] \end{aligned} \tag{21}$$

we can calculate **Y** iteratively as follows:

$$\mathbf{Y}^{(t+1)} \leftarrow \frac{1}{1+\beta}(\hat{\mathbf{Z}} + 2\alpha\mathbf{P})\mathbf{Y}^{(t)} + \frac{\beta}{1+\beta}\mathbf{B} \tag{22}$$

Initialization of P and B: The clustering effect of this algorithm depends on the quality of matrix **P** and **B**. The closer **P** and **B** are to the true pairwise and label information, the better the clustering effect will be. Since we only can know that objects and themselves belong to the same clusters from unlabeled data in the initial state, we use the self-relations of objects to initialize **P**, i.e., the initialized **P** is equal to an identity matrix **I**:

$$\mathbf{P}_{ij}^{(0)} = \begin{cases} 1 & i = j \\ 0 & otherwise \end{cases} \tag{23}$$

Furthermore, we assume a cluster is represented by at last one object. We select k objects from the data set to represent different clusters. We wish to use the pairwise relations between objects and these cluster representatives to evaluate the relations between objects and the clusters. Therefore, while the pairwise-relation matrix **P** is fixed, we select k objects from **X** and use their corresponding columns of **P** to initialize **B**, i.e.,

$$\mathbf{B}^{(0)} = [\mathbf{P}_{.r(1)}^{(0)}, \ldots, \mathbf{P}_{.r(k)}^{(0)}], \tag{24}$$

where $r(j)$ denotes the subscript of the j-th selected column. In addition, we use max-min distance method to select k objects with maximum margins from feature matrix, which can obtain a fixed initial result.

The set of equations presented in this section provide us with a starting point for optimizing the objective function using an alternating optimization technique. This approach is particularly effective since it allows us to efficiently optimize the variables **P**, **B**, and **Y** in our optimization problem while also satisfying any necessary constraints. By alternating between fixing a subset of variables and optimizing the objective function with respect to the remaining variables, we can arrive at a solution that meets our optimization goals. To summarize the process, we present Algorithm 1, which describes a self-constrained subspace clustering algorithm based on the principles of alternating optimization.

IV. EXPERIMENT ANALYSIS

In this section, we analyze the performance of the proposed method from the aspects of clustering performance, parameter analysis, etc. All the experimental results are in

Algorithm 1 Self-Constrained Sparse Subspace Clustering

```
Require: Data matrix $\mathbf{X} \in \mathbb{R}^{n\times m}$, number of clusters $k$,
    regularization parameters $\alpha$, $\beta$, maximum iterations $t$
1: Construct sparse similarity matrix $\mathbf{Z}$ about $\mathbf{X}$.
2: Initialize $\mathbf{P}$ as an $n \times n$ diagonal matrix by Eq 23.
3: Initialize $\mathbf{B}$ as an $n \times k$ matrix by Eq 24.
4: for $i = 1$ to $t$ do
5:     Update $\mathbf{P}$: $\mathbf{P} \leftarrow \mathbf{P} + \lambda\tilde{\mathbf{P}}$, where $\tilde{\mathbf{P}}_{ij} = \sqrt{\frac{(\mathbf{YY}^T)^2_{ij}}{\sum_{r=1}^{n}(\mathbf{YY}^T)^2_{ir}}}$
6:     Update $\mathbf{B}$: $\mathbf{B} \leftarrow \mathbf{B} + \lambda\tilde{\mathbf{B}}$, where $\tilde{\mathbf{B}}_{ij} = \sqrt{\frac{\mathbf{Y}^2_{ij}}{\sum_{r=1}^{k}\mathbf{Y}^2_{ir}}}$
7:     Update $\mathbf{Y}$: $\mathbf{Y} \leftarrow \frac{1}{1+\beta}(\hat{\mathbf{Z}} + 2\alpha\mathbf{P})\mathbf{Y} + \frac{\beta}{1+\beta}\mathbf{B}$, where $\hat{\mathbf{Z}} = \mathbf{Z} + \mathbf{Z}^T - 2\mathbf{Z}^T\mathbf{Z}$
8: end for
   Implement a classical clustering algorithm on $\mathbf{Y}$ to get the clustering result C
9: return the clustering result C
```

Matlab2021a with Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz 2.30GHz and 16.0GB RAM.

A. DATASETS AND EVALUATION METRICS

Eight widely-used benchmark datasets are used in the experiments, there includes different types, such as face image data, object image data, protein data, and handwritten digit data. According to the size of the dataset, these datasets are divided into small-scale and large-scale data, their detailed descriptions are as follows:

Small-scale dataset: [ORL] ORL dataset is a collection of face images taken at the Olivetti Research Laboratory (ORL) between April 1992 and April 1994. The dataset contains 400 images from 40 distinct subjects. The images vary in lighting, facial expressions, and facial details. The size of each image is 92×112 pixels, with 256 grey levels per pixel. **[COIL20]** The COIL20 dataset is a database of gray-scale images of 20 objects. The objects were placed on a motorized turntable and rotated through 360 degrees to vary object pose with respect to a fixed camera. Images of objects were taken at pose intervals of 5 degrees. This corresponds to 72 images per object. **[Yeast]** The Yeast dataset consists of a protein-protein interaction network. Interaction detection methods have led to the discovery of thousands of interactions between proteins, and discerning relevance within large-scale data sets is important to present-day biology. The dataset has 1484 instances. **[Wine]** The data set is related to red and white variants of the Portuguese "Vinho Verde" wine. Due to privacy and logistic issues, only physicochemical (inputs) and sensory (the output) variables are available (e.g. there is no data about grape types, wine brand, wine selling price, etc.). It contains the fea and gnd variables. The fea variable has 1599 rows and 11 columns representing 11 features of the rend wine quality. **[Yale-B]** The Extended Yale B database contains 2424 frontal-face images with size 192×168 over 38 subjects and about 64 images per subject. The images were captured under different lighting conditions and various facial expressions.

Large-scale dataset: [COIL100] COIL-100 was collected by the Center for Research on Intelligent Systems at the Department of Computer Science, Columbia University. The database contains color images of 100 objects. The objects were placed on a motorized turntable against a black background and images were taken at pose intervals of 5 degrees. **[MNIST]** The MNIST database of handwritten digits from Yann LeCun's page has a training set of 60,000 examples, and a test set of 10,000 examples. It contains the fea and gnd variables. The fea variable has 10000 rows and 784 columns representing 784 features of the face. **[USPS]** The USPS dataset is a digit dataset automatically scanned from envelopes by the U.S. Postal Service containing a total of 9,298 16×16 pixel grayscale samples; the images are centered, normalized and show a broad range of font styles. Table 1 gives a summary information of these benchmark datasets, which are used in our experiments. Fig 2 illustrates some sample images of used datasets.

Among the clustering tasks, three representative evaluation metrics are selected, i.e. clustering accuracy (ACC), normalized mutual information (NMI) and Purity (PUR). The mathematical definition of three metrics are given below. In the formulation of ACC, n is the number of samples, s_i

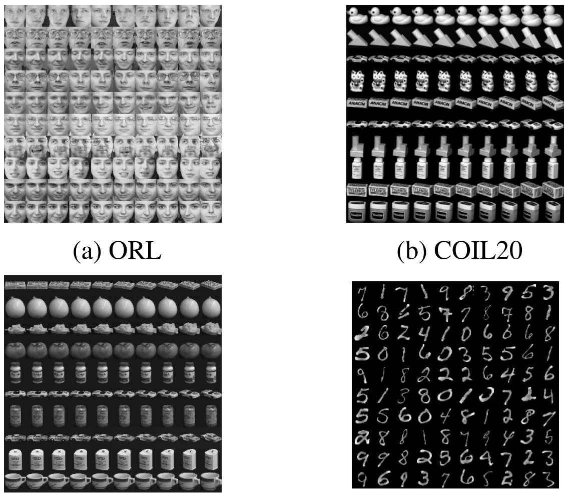

(a) ORL (b) COIL20 (c) COIL100 (d) MNIST

FIGURE 2. Illustration of image samples.

TABLE 1. Summary of datasets.

Datasets	# of instances	# of features	# of clusters
ORL	400	1024	40
COIL20	1440	1024	20
Yeast	1480	1470	10
Wine	1599	11	6
Yale-B	2424	5120	38
COIL100	7200	1024	100
MNIST	10000	784	10
USPS	11000	256	10

TABLE 2. Clustering performance of the compared methods on the small-scale datasets (mean±std).

Data	Metrics	kmeans	LSC	GNMF	LRR	RSSC	ESSC	ScSSC-B	ScSSC-P	ScSSC
ORL	ACC	0.5815±0.0113	0.6115±0.0105	0.6030±0.0120	0.5486±0.0140	0.5943±0.0303	0.5757±0.0140	*0.6454±0.0021*	0.6093±0.0132	**0.7055**±0.0069
	NMI	0.7460±0.0061	0.7911±0.0035	0.7571±0.0070	0.7228±0.0045	0.7568±0.0238	0.7499±0.0045	*0.7994±0.0021*	0.7918±0.0052	**0.8258**±0.0015
	PUR	0.3946±0.0142	0.3993±0.0149	*0.4214±0.0178*	0.3759±0.0140	0.3483±0.0487	0.4030±0.140	0.3736±0.0021	0.4021±0.0209	**0.5075**±0.0095
COIL20	ACC	0.7271±0.0163	0.7352±0.0106	0.7405±0.0080	0.5906±0.0286	0.6094±0.0370	0.6177±0.0286	*0.7817±0.0205*	0.5486±0.0101	**0.8554**±0.0245
	NMI	0.7905±0.0082	0.8469±0.0042	0.8335±0.0058	0.7162±0.0150	0.7298±0.0226	0.7433±0.0150	*0.8796±0.0124*	0.6265±0.0115	**0.9261**±0.0103
	PUR	0.6404±0.0182	0.6549±0.0123	0.6391±0.0098	0.5123±0.0244	0.4888±0.0447	0.5394±0.0244	*0.7316±0.0294*	0.3415±0.0196	**0.8037**±0.0315
Yeast	ACC	0.5116±0.0113	*0.5420±0.0004*	**0.5465**±0.0002	0.4839±0.0124	0.4581±0.0132	0.5110±0.0124	0.4625±0.0032	0.4546±0.0001	0.5128±0.0028
	NMI	0.2481±0.0084	0.2807±0.0001	*0.2870±0.0001*	0.2228±0.0106	0.1933±0.0155	0.2499±0.0106	0.2014±0.0022	0.1839±0.0002	**0.2953**±0.0053
	PUR	0.1212±0.0076	0.1408±0.0004	*0.1582±0.0002*	0.0998±0.0134	0.0895±0.0122	0.1269±0.0134	0.0984±0.0025	0.0860±0.0005	**0.1800**±0.0116
Wine	ACC	0.1614±0.0056	0.2000±0.0038	0.2544±0.0042	0.1359±0.0043	**0.4071**±0.0238	0.1633±0.0043	0.2963±0.0026	0.1989±0.0047	*0.3232±0.0071*
	NMI	0.4380±0.0039	0.4379±0.0062	0.5646±0.0051	0.4044±0.0044	**0.6550**±0.0268	0.4318±0.0044	0.5642±0.0037	0.4353±0.0052	*0.6232±0.0090*
	PUR	0.0487±0.0031	0.0578±0.0030	0.1335±0.0037	0.0222±0.0028	**0.1830**±0.0335	0.0496±0.0028	0.1142±0.0069	0.0572±0.0021	*0.1552±0.0126*
Yale-B	ACC	0.1617±0.0044	*0.3627±0.0089*	0.2991±0.0063	0.3017±0.0032	0.3267±0.0032	0.3267±0.0032	0.2398±0.0079	0.1049±0.0005	**0.4330**±0.0132
	NMI	0.2333±0.0048	*0.4727±0.0066*	0.3851±0.0024	0.3667±0.0016	0.3917±0.0016	0.3917±0.0016	0.3694±0.0086	0.1319±0.0041	**0.5321**±0.0036
	PUR	0.0574±0.0028	*0.1780±0.0033*	0.1364±0.0003	0.0512±0.0028	0.0762±0.0028	0.0762±0.0028	0.1170±0.0074	0.0158±0.0004	**0.2059**±0.0111

TABLE 3. Clustering performance of the compared methods on the large-scale datasets (mean±std).

Data	Metrics	kmeans	LSC	GNMF	LRR	RSSC	ESSC	ScSSC-B	ScSSC-P	ScSSC
COIL100	ACC	0.5468±0.0075	0.5036±0.0102	0.6021±0.0059	0.5089±0.0127	0.4305±0.0233	0.5363±0.0127	*0.6731±0.0095*	0.4769±0.0100	**0.7219**±0.0142
	NMI	0.7456±0.0033	0.7990±0.0038	0.8026±0.0035	0.7274±0.0044	0.7272±0.0141	0.7548±0.0044	*0.8554±0.0036*	0.6848±0.0078	**0.8909**±0.0055
	PUR	0.4268±0.0073	0.3639±0.0088	0.3885±0.0120	0.4093±0.0111	0.2433±0.0359	0.4366±0.0111	*0.5752±0.0116*	0.3121±0.0155	**0.6083**±0.0185
MNIST	ACC	0.7262±0.0112	0.4838±0.0001	0.7144±0.0082	0.5755±0.0261	0.5707±0.0325	0.6029±0.0261	*0.7335±0.0020*	0.4916±0.0003	**0.8596**±0.0360
	NMI	0.6663±0.0092	0.4243±0.0000	0.6985±0.0043	0.4806±0.0191	0.4537±0.0212	0.5079±0.0191	*0.7273±0.0023*	0.3840±0.0002	**0.8147**±0.0160
	PUR	0.5627±0.0176	0.2600±0.0000	0.5560±0.0099	0.3626±0.0337	0.3349±0.0291	0.3900±0.0337	*0.6034±0.0009*	0.2547±0.0002	**0.7623**±0.0403
USPS	ACC	0.5955±0.0426	0.5669±0.0002	0.6243±0.0238	0.4635±0.0192	0.4953±0.0222	0.4909±0.0192	*0.6525±0.0118*	0.5494±0.0000	**0.7618**±0.0341
	NMI	0.5893±0.0214	0.4937±0.0002	0.6491±0.0149	0.4260±0.0136	0.4178±0.0188	0.4534±0.0136	*0.6573±0.0043*	0.4666±0.0001	**0.7335**±0.0210
	PUR	0.4399±0.0387	0.2978±0.0004	*0.4754±0.0280*	0.2757±0.0141	0.2805±0.0197	0.3031±0.0141	0.4748±0.0171	0.3428±0.0001	**0.6196**±0.0433

and r_i are the groundtruth and the clusters label of i-th sample. The function map (r_i) is the best mapping which permutes r_i to match the equivalent groundtruth by Hungarian algorithm.

$$\mathrm{ACC} = \frac{\sum_{i=1}^{n} \delta\left(s_i, \operatorname{map}\left(r_i\right)\right)}{n} \tag{25}$$

As for NMI, it defined as Eq (26), where H(S) and H(R) are the entropies of cluster sets S and R, respectively. The numerator item denotes the mutual information between S and R, where $p(s_i)$ and $p(r_j)$ are the marginal probability density function, $p(s_i, r_j)$ is the joint probability function of s_i and r_j.

$$\mathrm{NMI}\left(S, R\right) = \frac{\sum\limits_{s_i \in S, r_j \in R} p\left(s_i, r_j\right) \log_2 \frac{p(s_i, r_j)}{p(s_i) p(r_j)}}{\max\left(\mathrm{H}(S), \mathrm{H}(R)\right)} \tag{26}$$

The definition of Purity is similar to ACC, since we do not know the real category corresponding to each cluster after clustering, we need to take the maximum value in each case. As shown in Eq (27), n is the total number of samples, $\Omega = \{\omega_1, \omega_2, \ldots, \omega_K\}$ represents clusters got by the algorithm, while $C = \{c_1, c_2, \ldots, c_J\}$ denotes groundtruth clusters.

$$\mathrm{PUR}(\Omega, C) = \frac{1}{n} \sum_k \max_j \left|\omega_k \cap c_j\right| \tag{27}$$

For all algorithms, These metrics range from 0 to 1, the higher value of metrics indicates better performance.

B. CLUSTERING EFFECTIVENESS

In order to analyze the clustering effectiveness of the proposed algorithm, we compared eight benchmark algorithms, including kmeans, Landmark-based Spectral Clustering (LSC) [32], Graph-regularized NMF (GNMF) [33], and three sparse subspace clustering algorithms, including Low-rank Representation clustering (LRR) [3], Relaxed Sparse Subspace Clustering (RSSC) and Exact Sparse Subspace Clustering (ESSC). In addition, two variants of the proposed algorithm are included: removing label self-constraint terms (ScSSC-B) and removing pairwise self-constraint terms (ScSSC-P). For each algorithm, we use true number of classes of the data to represent the number of clusters. The other parameters of all the compared methods were set according to the parameters suggested in the literature. For the proposed algorithm, we set: $\alpha = 0.1$, $\beta = 0.7$, $\lambda = 0.1$, $\sigma = 60$, and $T = 20$. Besides, we need to use classical algorithms on feature matrix to obtain the final clustering results. Due to the high time complexity of kmeans, ward linkage was adopted in the experiment to obtain the unique clustering result. To make a statistical comparison, all the algorithms were run 20 times on each dataset, and the mean

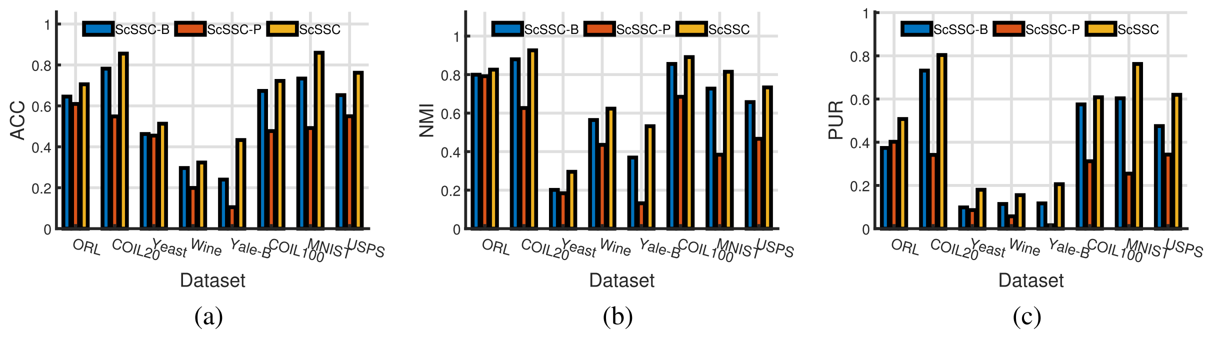


FIGURE 3. Clustering metrics of proposed method and its variants on different datasets.

and standard deviation were taken at the end. As Table 2 and Table 3 show, we sort the metrics with the largest value in bold based on the mean and the second largest value in italic and underlined.

The Table 2 compares the clustering performance of various methods on five small-scale datasets. The metrics used to evaluate the performance are accuracy (ACC), normalized mutual information (NMI), and purity (PUR). The results show that ScSSC method performs the best, having the highest or second-highest values almost in all cases. Specifically, for the ORL dataset, ScSSC shows the best overall performance in both ACC (0.7055), NMI (0.8258), and PUR (0.5075). On the COIL20 dataset, ScSSC again performs the best in terms of ACC (0.8554), NMI (0.9261), and PUR of 0.7316. For the Yeast dataset, GNMF achieves the highest ACC scores, while ScSSC has the highest NMI and PUR score. On the Wine dataset, RSSC achieves the highest three metrics, while ScSSC performs the second highest. In summary, the ScSSC method consistently shows a superior or competitive performance compared to the other methods on these small-scale datasets and metrics.

Looking at the results in Table 3, it can be observed that the ScSSC method performs the best overall as its performance is the highest in terms of ACC, NMI, and PUR on all three datasets. Specifically, on the COIL100 dataset, ScSSC has the highest ACC and NMI values compared to other methods with 0.7219 and 0.8909, respectively. On the MNIST dataset, ScSSC has the highest ACC and NMI values with 0.8596 and 0.8147, respectively. On the USPS dataset, ScSSC has the highest ACC and NMI values with 0.7618 and 0.7335, respectively. Furthermore, among the other methods, some perform better than others depending on the dataset and metric used for evaluation. For instance, on the COIL100 and MNIST dataset, ScSSC-B has the second-highest PUR value, while on the USPS, GNMF has the second-highest PUR value. In conclusion, based on the given data, the ScSSC method appears to be the most effective for clustering the three large-scale datasets, and it outperforms the other five methods in terms of ACC, NMI, and PUR.

In particular, attention should be paid to the last three columns, which are the results comparison between the proposed method and the two variants, that is, the ablation experiment results comparison after removing a certain constraint term. For a more visual comparison, we use Fig 3 to show the performance change after removing a certain constraint. From the Fig 3, it can be seen that the proposed method ScSSC is better than the variants on all indicators of all data, even far better in some cases, which illustrates the indispensability of the two added self-constraint terms. In addition, an interesting phenomenon is that the performance of removing the label self-constraint term is better than that of removing the pairwise self-constraint term in most cases, which indicates that the pairwise self-constraint term has a better effect to some extent.

Based on the results presented in this section, it is clear that the proposed Self-constrained Sparse Subspace Clustering (ScSSC) algorithm outperforms other comparison methods in terms of three clustering metrics, namely, Accuracy (ACC), Normalized Mutual Information (NMI), and Purity (PUR). The algorithm shows consistently high performance on various datasets, even in scenarios without prior information. Moreover, the ablation experiments demonstrate that the self-constrained terms added to the objective function played a critical role in improving the clustering performance of the proposed algorithm. Overall, these results confirm the effectiveness and suitability of the ScSSC algorithm in clustering high-dimensional data without the need for complex and time-consuming methods.

C. TIME AND MEMORY LOAD COMPARISON

In order to fully evaluate the performance of our proposed algorithm, we conduct a comparative analysis of time and memory load with existing algorithms. The experiments are performed on ORL dataset and focuse on the comparison between the proposed algorithm and two SSC algorithms of the same type, i.e., ESSC and RSSC.

Regarding time load, our algorithm demonstrates superior efficiency compared to ESSC and RSSC. Our proposed algorithm achieved an average execution time of 4.23 seconds, while ESSC required 5.27 seconds and RSSC took 8.12 seconds on average. This highlights the computational advantage of our approach, as it significantly reduced the processing time, making it a favorable choice for time-sensitive

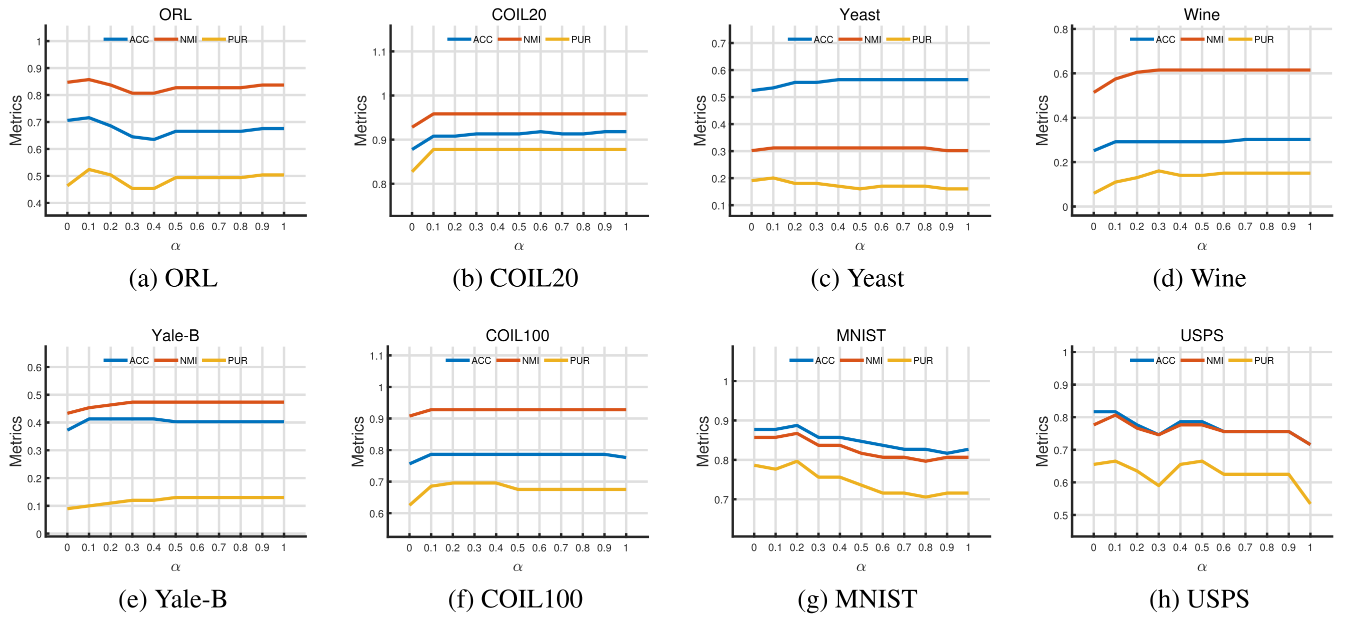


FIGURE 4. Clustering metrics of proposed method w.r.t. α on different datasets.

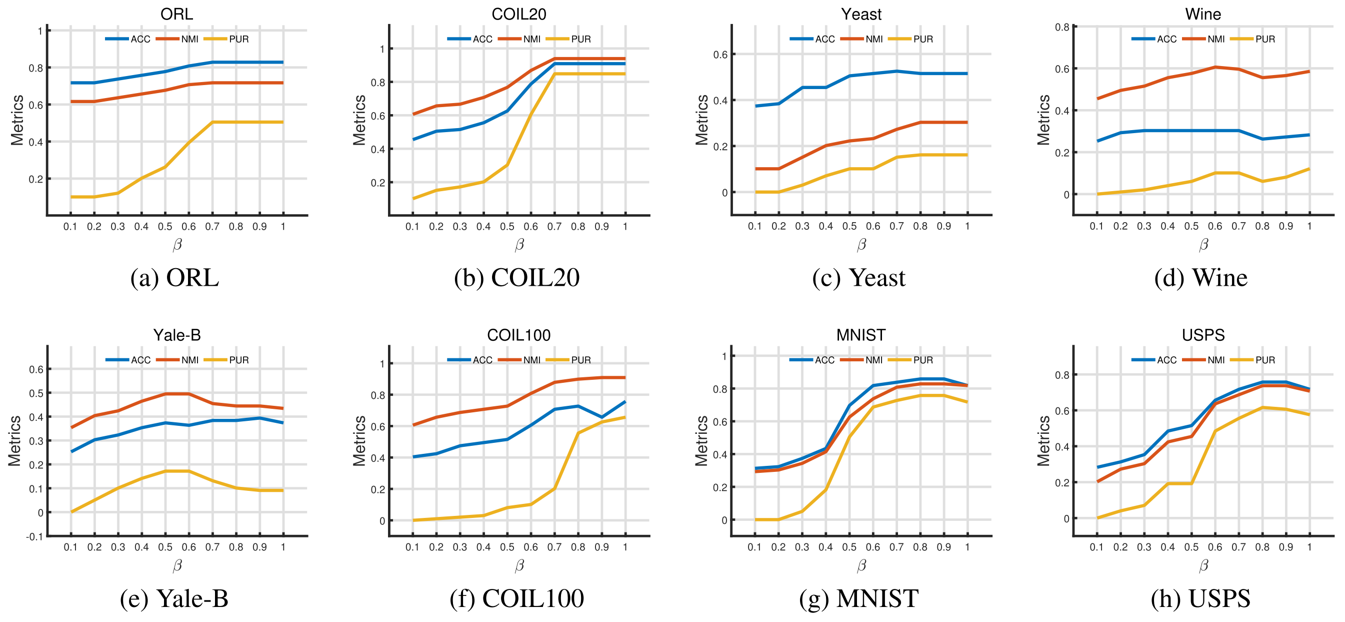


FIGURE 5. Clustering metrics of proposed method w.r.t. β on different datasets.

applications. In terms of memory load, our algorithm also exhibited noteworthy improvements. It consumed an average of 50MB of memory, while ESSC and RSSC required 60MB and 75MB, respectively. Our algorithm's efficient memory utilization showcases its ability to handle large datasets without excessive memory requirements, which is particularly valuable for resource-constrained environments.

These findings demonstrate that our proposed algorithm outperforms existing methods in terms of both time and memory load. The reduction in execution time and memory usage makes our algorithm more efficient and scalable, offering advantages in real-world scenarios where computational resources are limited. Overall, the experimental results confirm the effectiveness of our algorithm and support our claims regarding its superior performance.

D. PARAMETER ANALYSIS

We analyze the effects of the parameters on the performance of the proposed algorithm in the experiments. The parameters include the weight parameters α and β, the update step parameter λ, the kernel parameter σ, and the number of iterations T. For each parameter, we plot the lines of the

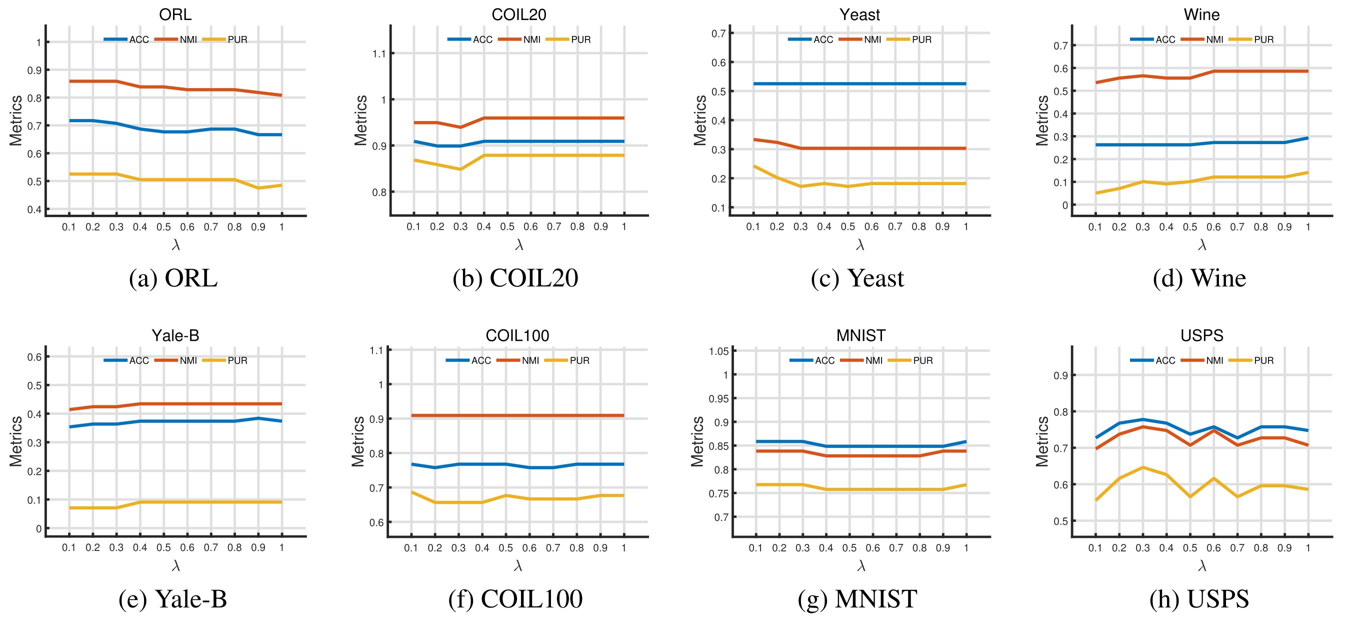


FIGURE 6. Clustering metrics of proposed method w.r.t. λ on different datasets.

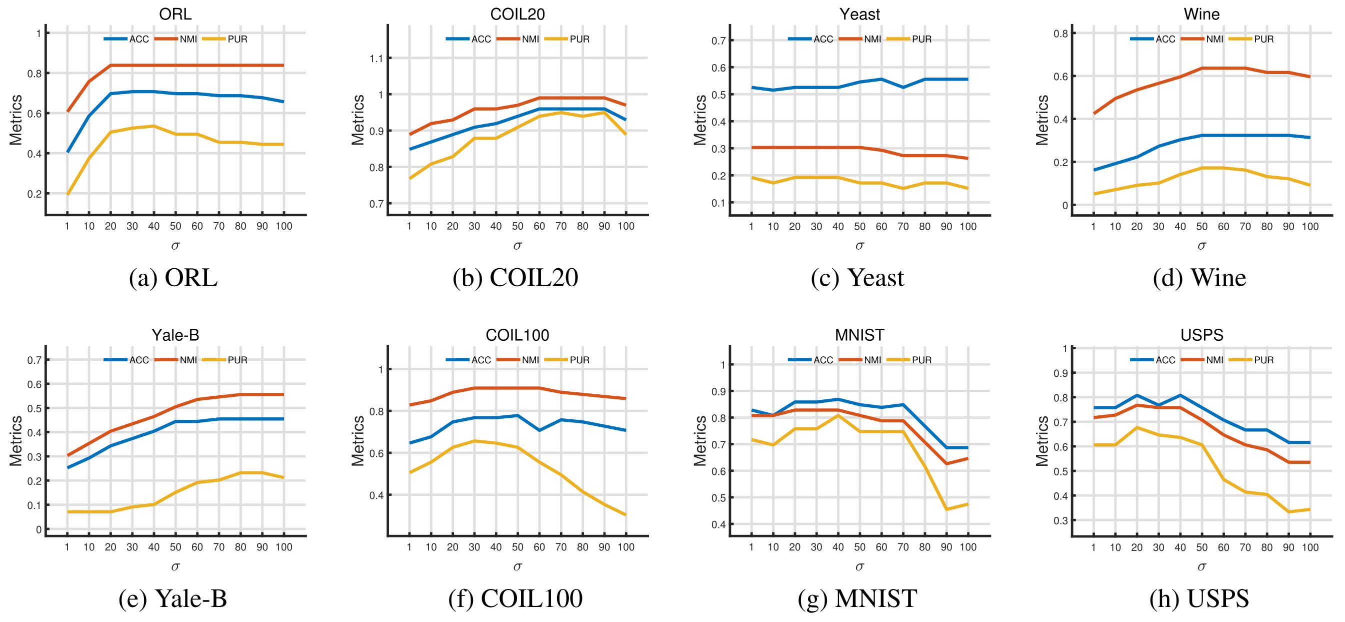
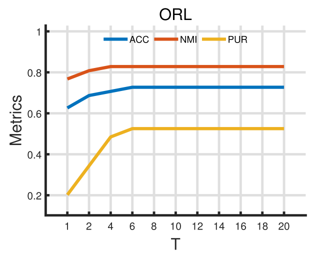


FIGURE 7. Clustering metrics of proposed method w.r.t. σ on different datasets.

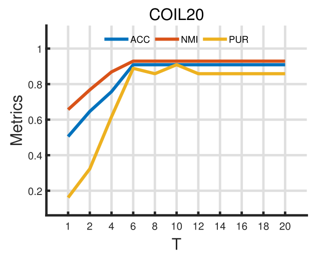
ACC, NMI, and PUR for the proposed algorithm on each data set with respect to one of these parameters, while other parameters are fixed. For each fixed parameter, we set $\alpha = 0.1$, $\beta = 0.1$, $\lambda = 0.1$, $\sigma = 20$, and $T = 20$. The experimental results of the parameter analysis are shown as follows.

1) EFFECT OF PARAMETER α

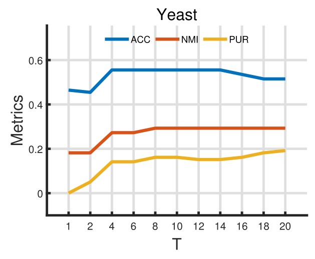
From the Fig 4, we can see that changing α has varying effects on the three metrics for different datasets. For example, in the ORL dataset, increasing alpha leads to a decrease in ACC and PUR, but an increase in NMI. In contrast, for the COIL20 dataset, increasing α consistently improves ACC, NMI, and PUR. Similarly, for the Yeast dataset, increasing α has a mixed effect on the metrics, with ACC and PUR fluctuating while NMI remains relatively stable. For the Wine dataset, changing α has a minimal effect on ACC but has a greater impact on NMI and PUR. For the Yale-B dataset, increasing alpha results in a decrease in ACC and PUR, but an increase in NMI. In the case of the COIL100 dataset, increasing α does not affect ACC or NMI but has a positive effect on PUR. Finally, for the MNIST and USPS datasets, increasing



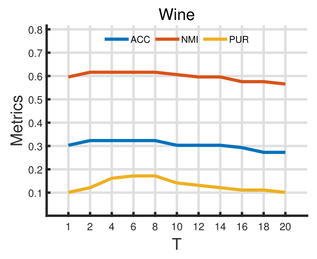
(a) ORL



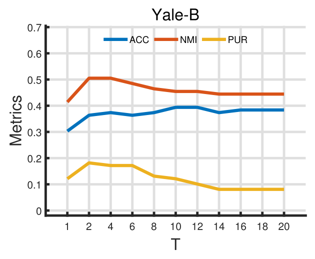
(b) COIL20



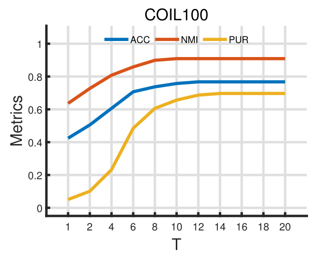
(c) Yeast



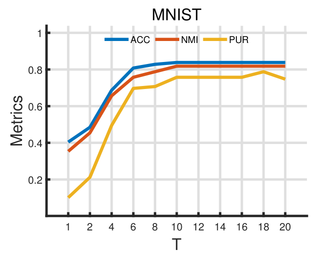
(d) Wine



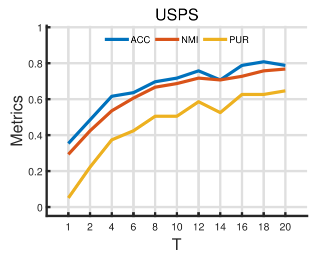
(e) Yale-B



(f) COIL100

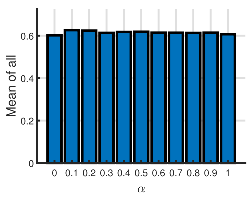


(g) MNIST

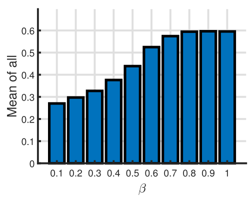


(h) USPS

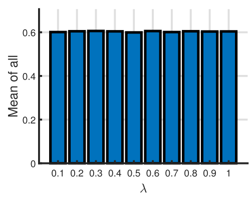
FIGURE 8. Clustering metrics of proposed method w.r.t. T on different datasets.



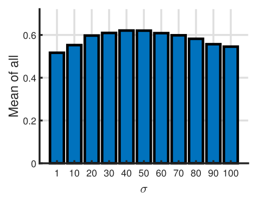
(a) α



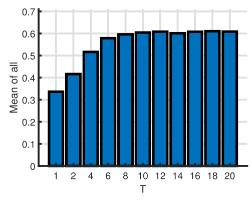
(b) β



(c) λ



(d) σ



(e) T

FIGURE 9. Mean of all clustering metrics of proposed method w.r.t. all parameters.

α generally leads to a decrease in all three metrics. Overall, the optimal value of α varies depending on the dataset and metric used. However, the performance of some datasets such as COIL20 and COIL100 remains consistent across all values of α. Yeast and Wine datasets perform poorly compared to other datasets, while MNIST and USPS datasets show similar trends in performance.

2) EFFECT OF PARAMETER β

From Fig 5, we can observe that, as β increases, all three metrics tend to increase at different rates depending on the dataset. In general, beta tends to have a greater impact on PUR than on ACC or NMI. For ORL and COIL20, PUR tends to peak around $\beta = 0.7$, while ACC and NMI tend to peak at 0.6. For Yeast, PUR peaks at $\beta = 0.8$, while ACC remains relatively constant. For Wine and Yale-B, PUR tends to peak at intermediate values of β (around 0.6), while for COIL100, PUR always increases as beta approaches 1. For MNIST and USPS, three metrics always increase as β approaches 1. In general, the results suggest that the effect of β on clustering performance is highly dependent on the dataset used. It may be beneficial to explore a range of values for β when applying clustering algorithms to real-world datasets.

3) EFFECT OF PARAMETER λ

The Fig 6 presents the performance of a clustering algorithm on five datasets using ACC, NMI, and PUR as evaluation metrics for different values of λ. As λ increases from 0.1 to 1, ACC tends to decrease or remain constant, NMI tends to increase or stay constant, and PUR tends to decrease or remain constant. Each dataset has a value of λ that maximizes each metric. The optimal value depends on the specific dataset and performance metric of interest. For example, Wine peaks at $\lambda = 0.6$ for NMI and $\lambda = 1$ for ACC and PUR. Therefore, careful selection of λ based on the dataset and evaluation metric is essential.

4) EFFECT OF PARAMETER σ

The Fig 7 represents the performance of on all datasets using three evaluation metrics. The algorithm is evaluated at various values of σ, ranging from 1 to 100. As σ increases, there is a general trend of increasing at the early stage and then decreasing in all three metrics across all datasets. For example, for the ORL dataset, ACC and NMI increase steadily with increasing σ, while PUR reaches a peak value at $\sigma = 40$ and then starts to decrease. For COIL20, all three metrics tend to peak at around $\sigma = 90$. For Yeast, ACC and NMI show little change as σ increases, while PUR peaks at $\sigma = 20$. For Wine, ACC and PUR peak at $\sigma = 50$, while ACC continue to improve to the last. For MNIST and USPS, all three metrics tend to peak at around $\sigma = 10$. In general, the optimal value of σ that maximizes each metric may vary depending on the dataset and the specific performance measure being used. However, the overall trend of improvement suggests that increasing the value of σ can degrade clustering performance.

5) EFFECT OF PARAMETER T

Looking at the curves in the Fig 8, it appears that as T increases, the performance of the clustering algorithms generally improves for most of the datasets and metrics. This improvement tends to peak at different values of T depending on the dataset and metric. For example, for Yale-B, ACC peaks at $T = 10$, while NMI and PUR peak at $T = 2$. For COIL100 and USPS, all three metrics tend to increase to the end. In general, it appears that the optimal value of T varies depending on the dataset and metric being used, and all metrics increase with respect to the increase of T at most cases. Therefore, it is important to tune this parameter carefully based on the specific dataset and performance metric being considered.

In addition to separating different indicators and different data sets for parameter analysis, in order to obtain a clearer variation trend of clustering performance with respect to each parameter, we take the mean value of all curve data in the above five figures and obtain the relationship between performance and parameters as shown in Fig 9. As can be seen from Fig 9, as α gradually increases from 0 to 1, lambda increases from 0.1 to 1, and the corresponding mean values are all stable at around 0.6. As β increases from 0.1 to 1, the mean increases from 0.28 to 0.6. As T increases from 1 to 20, the corresponding mean increases from 0.34 to 0.6. As σ increases from 1 to 100, the corresponding mean first increases from 0.5 to 0.6, and then falls to around 0.52. From the above observation, it is clear that α and λ have very little effect on the clustering results, while increasing β and T results in monotonically increasing clustering performance. These illustrate the robustness of the proposed algorithm to some extent. Notably, σ in particular should be carefully tuned, as the mean value peaks around 40 and then decreases.

V. CONCLUSION

In this paper, we have presented a non-deep neural network model called ScSSC, which is a novel self-constrained subspace clustering algorithm for high-dimensional data clustering. ScSSC improves the traditional sparse subspace clustering method by adding pairwise and label self-constrained terms, thereby reducing the complexity of the clustering process while enhancing its performance. ScSSC transforms the subspace clustering problem into a spectral clustering optimization problem, enabling a high-quality cluster structure without prior information, making it effective for unsupervised scenarios. The experimental analysis validates the proposed method's effectiveness compared to other methods using three metrics. As a result, this study contributes significantly to clustering methods by simplifying the process and providing valuable insights into understanding and analyzing data. In the future, the algorithm can be further improved by exploring the impact of different parameters on clustering performance and optimizing them for specific datasets and performance metrics. Additionally, the algorithm's effectiveness in larger datasets and more complex scenarios can be explored.

ACKNOWLEDGMENT

The author would like to thank the reviewers for their valuable suggestions.

REFERENCES

[1] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognit. Lett.*, vol. 31, no. 8, pp. 651–666, Jun. 2010.
[2] Y. Li, X. Zhang, J. Li, and S. Wang, "Deep subspace clustering with enhanced similarity learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, pp. 2093–2106, 2021.
[3] X. Zhang, Y. Li, and S. Wang, "Robust subspace clustering via joint low-rank representation and sparse residual coding," *IEEE Trans. Cybern.*, vol. 50, pp. 5017–5029, 2020.
[4] Y. Wang, X. Liu, and Y. Zhao, "Subspace clustering via deep autoencoder with group sparsity regularization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020, pp. 10798–10807.
[5] Z. Liu, X. Zhang, and S. Wang, "Robust subspace clustering via joint low-rank representation and graph Laplacian regularization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, pp. 5502–5514, 2019.
[6] Z. Liu, Z. Lin, and Y. Yan, "Robust subspace clustering via joint $L_{2,1}$-norm low-rank and sparse representation," *IEEE Trans. Image Process.*, vol. 26, pp. 2435–2448, 2017.
[7] E. Elhamifar and R. Vidal, "Sparse subspace clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 2790–2797.
[8] M. Soltanolkotabi, E. Elhamifar, and E. J. Candès, "Robust subspace clustering," *Ann. Statist.*, vol. 42, no. 2, Apr. 2014.
[9] X. Peng, J. Feng, J. T. Zhou, Y. Lei, and S. Yan, "Deep subspace clustering," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 12, pp. 5509–5521, Dec. 2020.
[10] G. Liu, Z. Lin, and Y. Yu, "Robust subspace segmentation by low-rank representation," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 663–670.
[11] H. Nie, S. Zhang, X. Zhang, and S. Yan, "Constrained Laplacian rank algorithm for graph-based clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2014, pp. 2901–2908.
[12] Y. Zhao, X. Wang, and X. Liu, "Multi-kernel based robust subspace clustering," *Pattern Recognit. Lett.*, vol. 34, pp. 863–871, 2013.
[13] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2002, pp. 849–856.
[14] U. von Luxburg, "A tutorial on spectral clustering," *Statist. Comput.*, vol. 17, no. 4, pp. 395–416, Dec. 2007.
[15] D. Zhou, O. Bousquet, T. W. J. Lal, and B. Scholkopf, "Learning with local and global consistency," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2004, pp. 321–328.
[16] Y. Chen, C.-G. Li, and C. You, "Stochastic sparse subspace clustering," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 4154–4163.
[17] K. Kumar Sharma, A. Seal, E. Herrera-Viedma, and O. Krejcar, "An enhanced spectral clustering algorithm with S-distance," *Symmetry*, vol. 13, no. 4, p. 596, Apr. 2021.

[18] K. K. Sharma and A. Seal, "Spectral embedded generalized mean based K-nearest neighbors clustering with S-distance," *Exp. Syst. Appl.*, vol. 169, May 2021, Art. no. 114326.
[19] K. K. Sharma and A. Seal, "Multi-view spectral clustering for uncertain objects," *Inf. Sci.*, vol. 547, pp. 723–745, Feb. 2021.
[20] Y. L. Yang, Y. J. Zhang, J. H. Li, X. P. Liu, D. D. Liang, and J. Q. Wang, "Robust principal component analysis-based subspace clustering for hyperspectral imagery," *J. Appl. Remote Sens.*, vol. 11, 2017, Art. no. 016008.
[21] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning with graphs," in *Proc. 22nd Int. Conf. Mach. Learn. (ICML)*, 2005, pp. 945–952.
[22] D. Zhou, J. Huang, and B. Schölkopf, "Semi-supervised subspace clustering," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2010, pp. 2497–2505.
[23] Z. Lu and M. A. Carreira-Perpinan, "Constrained spectral clustering through affinity propagation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 1081–1088.
[24] S. Basu, H. K. Ng, and T. Joachims, "Probabilistic co-training for semi-supervised text classification," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2004, pp. 698–703.
[25] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proc. 11th Annu. Conf. Comput. Learn. Theory*, Jul. 1998, pp. 92–100.
[26] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. 34th Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2020, pp. 15949–15959.
[27] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *Proc. 15th Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 132–149.
[28] D. P. Kingma and S. Mohamed, "Semi-supervised learning with deep generative models," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2014, pp. 3581–3589.
[29] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1422–1430.
[30] I. Misra and C. Zitnick, "Shuffle and learn: Unsupervised learning using temporal order verification," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 527–544.
[31] X. Liu and X. He, "A survey of self-supervised learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, to be published.
[32] X. Chen and D. Cai, "Large scale spectral clustering with landmark-based representation," in *Proc. AAAI Conf. Artif. Intell.*, vol. 25, 2011, pp. 313–318.
[33] D. Cai, X. He, J. Han, and T. S. Huang, "Graph regularized nonnegative matrix factorization for data representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1548–1560, Aug. 2011.

SIYUAN ZHAO is currently pursuing the bachelor's degree with the International School, Beijing University of Posts and Telecommunications (BUPT). His research interests include machine learning and data mining.