A Group Norm Regularized Factorization Model for Subspace Segmentation

XISHUN WANG1, (Member, IEEE), ZHOUWANG YANG2, XINGYE YUE1, AND HUI WANG3

1Center for Financial Engineering, Soochow University, Suzhou 215006, China
2School of Mathematical Sciences, University of Science and Technology of China, Hefei 230026, China
3State Grid Corporation Customer Service Center, Big Data Service Department, Nanjing 210000, China

Corresponding author: Xishun Wang (xswang11@stu.suda.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 11971342.

ABSTRACT Subspace segmentation assumes that data comes from the union of different subspaces and the purpose of segmentation is to partition the data into the corresponding subspace. Low-rank representation (LRR) is a classic spectral-type method for solving subspace segmentation problems, that is, one first obtains an affinity matrix by solving a LRR model and then performs spectral clustering for segmentation. This paper proposes a group norm regularized factorization model (GNRFM) inspired by the LRR model for subspace segmentation and then designs an Accelerated Augmented Lagrangian Method (AALM) algorithm to solve this model. Specifically, we adopt group norm regularization to make the columns of the factor matrix sparse, thereby achieving a purpose of low rank, which means no Singular Value Decompositions (SVD) are required and the computational complexity of each step is greatly reduced. We obtain affinity matrices by using different LRR models and then performing cluster testing on different sets of synthetic noisy data and real data, respectively. Compared with traditional models and algorithms, the proposed method is faster and more robust to noise, so the final clustering results are better. Moreover, the numerical results show that our algorithm converges fast and only requires approximately ten iterations.

INDEX TERMS Low-rank representation, group norm regularization, subspace segmentation, affinity matrix, spectral clustering.

I. INTRODUCTION

With the advent of the era of big data, we are confronted with a great deal of data every day. Although the data volume is large, it may only come from several low-rank subspaces. Subspace segmentation [1] divides data into several clusters and each cluster is a subspace; this problem arises in machine learning, computer vision, image processing, finance, and other fields [2]–[6]. Subspace segmentation is an important clustering problem, which is mainly solved by the following four methods: mixture of Gaussian [7], factorization [8], algebraic [9], and spectral-type methods [1], [10].

In the spectral-type method, one first calculates an affinity matrix and then spectral clustering (such as Normalized Cuts (N_Cut) [11]) is performed. The main variation among spectral-type methods is the different affinity matrix learning methods. Spectral clustering has attractive advantages, for example the algorithm is efficient, the data can be of any shape, the method is not sensitive to abnormal data, and can be applied to high-dimensional problems. Recently there have been many studies and developments in spectral clustering, such as [12]–[17].

LRR is a classic, effective spectral-type method for solving subspace segmentation problems. In 2010, the Low-Rank Representation (LRR) [1] problem was proposed by Liu et al. They assume that data samples come from the union of multiple subspaces, and the purpose of the LRR method is to denoise and obtain samples from the corresponding subspaces to which they belong. And they proved that LRR accurately obtains each real subspace for clean data. For noisy data, LRR approximately restores the subspace of the original data with theoretical guarantees. In reference [1], the spectral clustering performance using the affinity matrix obtained by LRR is more accurate and robust than other methods.

When solving the LRR problem, traditional methods tend to minimize the nuclear norm to approximate the minimum rank in the objective function. This is a convex approximation
that guarantees convergence in the designed algorithm. However, singular value decomposition (SVD) is required in order to solve nuclear norm problems. An SVD is time-consuming, and the computation complexity is \( O(n^3) \) for an \( n \times n \) affinity matrix. Many classic algorithms employ SVD to solve LRR, such as the Accelerated Proximal Gradient method (APG [18]), Alternating Direction Method (ADM [19]), and Linearized Alternating Direction Method with Adaptive Penalty (LADMAP [20]). APG solves an approximated problem of LRR, but the clustering results are inferior. LADMAP performs best among these algorithms; however, the calculation speed is still slow, especially for high-dimensional data. Along this line of thinking, accelerated LADMAP (LADMAP(A) [20]) is proposed by Lin et al. They used skinny SVD technology to reduce the complexity to \( O(m^2) \), where \( r \) is the rank of the affinity matrix. However, the rate of convergence is sub-linear, which requires more iterations, and the rank depends on hyperparameter selection. Lu et al. introduced a smooth objective function with regularization terms and used the Iterative Reweighted Least Squares (IRLS) method to solve objective functions [21]. This new method does not need SVD, but the computation complexity of matrix multiplication is \( O(n^3) \). Their numerical experiments show that the convergence is linear, so it is faster than LADMAP(A) in some cases.

In order to avoid the need for SVD calculation, Chen et al. offered the matrix factorization LRR model and Hidden Matrix Factors Augmented Lagrangian Method (HMFA) [22]. They decomposed an affinity matrix into \( UV \) and then used the Augmented Lagrangian Method (ALM) to solve the model, where \( U \in \mathbb{R}^{n \times r} \), \( V \in \mathbb{R}^{d \times n} \). Factorization model is difficult to determine rank, so Chen et al. designed a greedy method to traverse the rank \( r \), that is, they first choose a proper interval \( d \) and run the algorithm on the ranks \( 1, d + 1, 2d + 1, \ldots, k d + 1, \ldots \), and stop when the results begin to worsen. Thus, this process searches through the options one by one to find the optimum rank. Although the original problem becomes non-convex, the algorithm does not require SVD; only multiplication of the factor matrix is required. Its complexity is \( O(rmn) \), where \( m \) is the dimension of the data.

HMFA requires an outer loop to find rank \( r \), the inner loop iterates to meet the stopping criterion, and rank-finding result is heavily dependent on the given hyperparameter. To overcome the shortcomings of HMFA, we introduce group norm regularization of \( U \) to design an adaptive rank-finding matrix factorization model (GNRFM) to solve LRR. We first let \( U \in \mathbb{R}^{n \times K} \), where \( K \) is a larger number. Group norm regularization \( l_{2,1} \) makes some columns of the factor matrix \( U \) become zero vector columns. In this manner, the rank of the affinity matrix is automatically reduced to achieve the purpose of adaptively adjusting the rank. We design the Accelerated Augmented Lagrangian Method (AALM) algorithm to solve GNRFM. In summary, the contributions of this work include:

- We first use the group norm regularization method to solve the rank minimum problem. Specifically, we design the GNRFM model to solve the subspace segmentation problem. Compared with the traditional nuclear norm LRR model, our model has less computational complexity. Compared with the factorization model HMFA [22], the GNRFM model can adaptively find ranks without greedy searches.
- The group regularization term also has positive anti-noise effects, so the GNRFM is more robust.
- We design the AALM algorithm to solve GNRFM, which uses the acceleration technique of one-step inner iteration and deletes the zero vector column to reduce the computational complexity in each step. The numerical results show that the AALM algorithm converges in about ten steps.

The structure of this paper is as follows. Section 2 introduces the LRR problem, its nuclear norm approximation model [1], and the matrix factorization model [22]. Section 3 introduces our model, GNRFM, details the Accelerated ALM (AALM) for our model, gives time complexity analysis and introduces how to use GNRFM’s solution for spectral clustering. Numerical experimental results are reported in Section 4. Finally, Sections 5 concludes this paper.

II. THE LRR PROBLEM AND TWO TYPES OF MODELS

In order to facilitate a shared understanding of notations, we offer a summary in Table 1 of the primary notations used in this paper.

| \( Z \) | The solution to the LRR problem |
| \( W \) | The affinity matrix |
| \( X \) | The data matrix |
| \( E \) | The LRR problem’s error matrix |
| \( U \) | The factor matrix of factorization models HMFA and GNRFM |
| \( V \) | The factor matrix of factorization models HMFA and GNRFM |
| \( m \) | The dimension of a data matrix |
| \( n \) | The number of data samples |
| \( r \) | The rank of \( Z \) (columns \( U \) and row \( V \)) |
| \( K \) | The oversize rank estimate |
| \( \mu \) | The regularized parameter in the LRR and HMFA models |
| \( \mu_U \) | The group norm regularized parameter in GNRFM |
| \( \mu_Y \) | The Frobenius norm regularized parameter in GNRFM |
| \( Y \) | The multiplier matrix in the ALM and AALM algorithms |
| \( \beta \) | The penalty parameter in the ALM and AALM algorithms |
| \( t \) | The outer iteration step |
| \( i \) | The inner iteration step |
| \( \sigma \) | The noise intensity |
| \( k \) | The number of clusters |
| \( \| Z \|_F \) | The Frobenius norm |
| \( \| Z \|_{l_{2,1}} \) | The group norm of \( Z \) |
| \( x_i \) | The \( i \)-th column of a matrix |
| \( y_i \) | The \( i \)-th row of a matrix |

First, let us recall the following LRR problem:

\[
\min_{Z} \text{rank}(Z), \quad \text{s.t.} \ X = ZX, \quad (1)
\]

where \( X \in \mathbb{R}^{n \times n} \) is the data matrix, \( m \) is the dimension of the data vector, \( n \) is the number of data vectors, and \( Z \in \mathbb{R}^{n \times n} \). We refer to the optimal solution \( Z^* \) of the above problem as
the “lowest-rank representation” of data \(X\) with respect to a dictionary \(X\). This is an NP-hard problem because the rank is \(l_0\) norm, and the solution is not unique. As in the classic method for solving low-rank problems, Liu et al. [1] took advantage of the nuclear norm to approximate and obtain the following convex optimization problem:

\[
\min_{Z} ||Z||_*, \quad \text{s.t.} \ X = XZ. \tag{2}
\]

Liu et al. [23] proved that under some conditions, the solution of (2) is unique and is one of the solutions to (1). This solution \(Z^*\) can be transformed to obtain an affinity matrix for data \(X\), which can then be used for subspace segmentation. The uniqueness of (2) is given by Siming and Zhouchen [24]:

**Theorem 1:** Suppose the skinny SVD of \(X\) is \(X = \hat{U} \Sigma \hat{V}\), then the minimizer to problem (2) is uniquely defined by

\[
Z^* = \hat{V} \hat{V}^T. \tag{3}
\]

This formula naturally implies that \(Z^*\) is precisely able to recover an affinity matrix in [25].

Since the solution of (2) is one of the solutions for (1), we recommend referring to corollary 4.1 in [23]. To make the model robust to noise, Liu et al. [23] proposed the following noisy LRR nuclear norm model:

\[
\min_{Z,E} ||Z||_* + \mu ||E||_{2,1}, \quad \text{s.t.} \ X = XZ + E, \tag{4}
\]

where \(E \in \mathbb{R}^{m \times n}, ||E||_{2,1} = \sum_i \sqrt{\sum_j (E_{ij})^2}\) is the group norm of \(E\).

In order to solve (4), several algorithms based on SVD were designed, which are lack of speed. So Chen et al. [22] put \(Z\) into a low rank factorization \(Z = \hat{U} \hat{V}\), and they proposed the following matrix factorization model:

\[
\min_{\hat{U},\hat{V},E} ||E||_{2,1} \quad \text{s.t.} \ X = X\hat{U} \hat{V} + E. \tag{5}
\]

where \(\hat{U} \in \mathbb{R}^{m \times K}, \hat{V} \in \mathbb{R}^{n \times R}\). If we write \(U = X\hat{U}\), then the model is expressed as follows:

\[
\min_{U,V,E} ||E||_{2,1} \quad \text{s.t.} \ X = UV + E. \tag{6}
\]

However, the rank \(r\) of this model must be specified. Chen et al. [22] designed a greedy method to find the optimal rank:

1. Provide the interval \(d\) and hyperparameter \(\mu\).
2. Solve the problem (6) when \(r = 1, d + 1, 2d + 1, \ldots, kd + 1, \ldots\) and stop when \(r + \mu||E(r)||_{2,1}\) begins to worsen. Thus, they search through the options one by one to find the optimum rank.

If we assume the optimal rank \(r = r^\star\), in this case, the solution obtained from (6) is \((U^\star, V^\star, E^\star)\). According to the fact that data space \(X\) is full and the theorem in ([1], [23]), we obtain the optimal \(Z^\star\) by \(Z^\star = X^\star U^\star V^\star\) (\(X^\star\) is the pseudoinverse of \(X\)). The obtained rank is heavily dependent on the hyperparameter \(\mu\), and numerous additional iterative calculations must be done before the optimal rank is obtained. In order to find the rank adaptively to reduce the number of iterations, we design a new model in section 3, which adds the group norm regularization term \(||U||_{2,1}\) to the model (6).

### III. THE GROUP NORM REGULARIZED FACTORIZATION MODEL AND ALGORITHM

For calculating speed, the matrix factorization method is superior to the nuclear norm approximation method. However, it is difficult to estimate the rank of the restored matrix \(Z\) using the former method. So, our goal is to find an adaptive method for estimating the ranks of different types of data. The rank of a matrix is determined by the number of rows or columns in the factor matrix, and the rank of the matrix is reduced if some columns are zero. So, we first take an oversized factor matrix, and make the number of columns in the factor matrix zero by introducing group norm regularization; this achieves the purpose of adjusting the rank adaptively.

#### A. THE GROUP NORM REGULARIZED FACTORIZATION MODEL

If we assume that \(X \in \mathbb{R}^{m \times n}\) is a matrix of data samples, \(m\) is the dimension of the data, \(n\) is the number of data points, and some data contain noise. We hope to remove noise and represent clean data with a low rank to obtain an affinity matrix. We obtain the group norm regularized factorization model (GNRFM) by adding the group norm regularization term \(||U||_{2,1}\) to (6):

\[
\min_{U,V,E} ||E||_{2,1} + \mu ||U||_{2,1} + \frac{\mu \nu}{2} ||V||_F^2 \quad \text{s.t.} \ X = UV + E, \tag{7}
\]

where \(X \in \mathbb{R}^{m \times n}, U \in \mathbb{R}^{m \times K}, V \in \mathbb{R}^{K \times n}\). \(K\) is a larger number, and \(||.||_F\) is classic Frobenius norm. \(||U||_{2,1}\) is the group norm of \(U\), and \(||U||_{2,1} = \sum_i \sqrt{\sum_j (U_{ij})^2}\). The true rank \(r\) of \(X\) is usually unknown, and \(K\) is a relatively large initial guess, such as \(K = n\). Owing to the group norm regularization, some columns in \(U\) will be equal to zero under proper parameters \(\mu_U, \mu_V\). Assuming \(s\) columns in \(U\) will equal zero based on the group norm \(||.||_{2,1}\), then we can get \(\text{rank}(UV) = K - s\). So, we reach the goal of adjusting the rank of \(UV\) adaptively only by introducing group norm regularization. \(||V||_F^2\) is also very important because \(U\) and \(V\) play roles of balance and mutual restraint in GNRFM.

In summary, the GNRFM model adaptively estimates ranks for different types of data without the need to design additional updated rank strategies. The regularization terms make the model more resistant to noise. Of course, we introduce two extra hyperparameters \(\mu_U, \mu_V\), but numerical results show that our model is less sensitive to hyperparameters relative to other models.

#### B. THE AUGMENTED LAGRANGIAN METHOD (ALM)

In this section, we introduce the ALM method to solve (7). For such bi-convex problems, i.e., convex in \(U\) and convex in \(V\) for a fixed \(U\), Sun and Fevotte [26], Shen et al. [27], Xu et al. [28], and Chen et al. [22] all used similar ALM methods to solve such bi-convex problems, and
obtained relatively good numerical results. The augmented
Lagrange function in formula (7) is defined as follows:
\[
L_{\beta_t}(U, V, E, Y) = \|E\|_2 + \mu_U \|U\|_2 + \frac{\mu_V}{2} \|V\|_F^2 + \beta \|U V + E - X\|_F^2,
\]
where \( \beta \) is a penalty parameter, \( Y \in \mathbb{R}^{m \times n} \) is the Lagrange multiplier corresponding to the constant \( X = U V + E \) and 
\[
(U_t^{l+1})_{i} = \max(||(Q_{si})||_2 - \frac{\mu U}{\beta_t * \xi(V_t^l)}, 0) \frac{Q_{si}}{||Q_{si}||_2}.
\]
We obtain the solution to (14) by soft threshold shrinkage:
\[
(U_{t+1}^{l+1})_{si} = \max(||(Q_{si})||_2 - \frac{\mu U}{\beta_t * \xi(V_t^l)}, 0) \frac{Q_{si}}{||Q_{si}||_2},
\]
where \( Q_{si} = (U_t^l - (U_t^l V_t^l + E_t - X + Y_t^l) V_t^l)^T / \xi(V_t^l) \), \( i = 1, 2, \ldots, K \), and \( X_{si} \) represents the \( i \)-th column of \( X \).

Owing to the soft-thresholding rule, some columns in \( U \) are equal to zero, so we obtain a low-rank solution. Similarly, we get an explicit expression of \( E \):
\[
(E_{t+1})_{si} = \max(||(X - U_{t+1} V_{t+1} - Y_t / \beta_t)_{si}||_2 - \frac{1}{\beta_t}, 0)
\]
\[
\times (||X - U_{t+1} V_{t+1} - Y_t / \beta_t)_{si}||_2, \quad (16)
\]
where \( i = 1, 2, \ldots, n \).

To avoid ALM converging to an infeasible point, we adopt the strategies proposed by Lu and Zhang [29] to update \( \beta_t \) in the third part of Algorithm 1. At this point, we have given the explicit formula for updating the variables in (9) at the \( t \)-th iteration. According to the above updating formula, we employ Algorithm 1 to solve problem (7).

**Algorithm 1 ALM for GNRFM**

**Input:** Data \( X \), an overestimated rank \( K \), and hyperparameters \( \mu_U, \mu_V \);

**Initialize:** \( U_0 \in \mathbb{R}^{m \times K}, V_0 \in \mathbb{R}^{K \times n} \) are obtained by the SVD of \( X \) \( (U_0 = \tilde{U}, V_0 = \tilde{V}) \), \( Z_0 = Y_0 = 0, \beta_0, \rho > 1, \xi \in (0, 1), v \in (0, 1), (e_i) \geq 0, \) with \( \lim_{t \to \infty} e_t = 0, \) and a sufficiently large constant \( T > \max \{f(x, L_{\beta_t}(x, v)) \} \) and \( t = 0; \)

**while** not convergent do

1. With \( Y_t = Y_t \) and \( \beta = \beta_t \), compute \( V_{t+1}, U_{t+1} \) according to (13) and (15) to find an approximate solution for (10). Then, update \( E_{t+1} \) according to (16), so we can find an approximate point \( x_{t+1} = (U_{t+1}, V_{t+1}, E_{t+1}) \) s.t.
\[
||\nabla_x L_{\beta_t}(x, Y_t)|| \leq \epsilon_{t+1}, \quad L_{\beta_t}(x_{t+1}, Y_t) < T; \quad (17)
\]

2. Set \( Y_{t+1} = Y_t + \beta_t (U_{t+1} V_{t+1} + E_{t+1} - X) \); (18)

3. If \( t > 0 \) and \( ||U_{t+1} V_{t+1} + E_{t+1} - X|| \leq \xi \|U_t V_t + E_t - X\| \), then set \( \beta_{t+1} = \beta_t \), Otherwise, set \( \beta_{t+1} = \max(\rho \beta_t, ||Y_{t+1}||_1^{1+}) \); (20)

4. Set \( t \leftarrow t + 1 \).

**end while**

Many books or articles (Boyd [30], Chen et al. [22], Sun and Fevotte [26], Shen et al. [27], Xu et al. [28]) all numerically show strong convergence behaviour and fast calculation.
speed for non-convex problems like this type of matrix factorization. But the proof of convergence about applying the ALM to non-convex problems is still a very difficult matter at present. The last four articles assume that ALM algorithm converges to the KKT point under some strong conditions, which are difficult to verify theoretically. Thus this topic is deserving of future research. For a detailed discussion of convergence, we recommend readers to see [22].

C. THE ACCELERATED ALM METHOD (AALM) FOR GNRFM

In this section, we propose two techniques to accelerate the ALM for GNRFM. The techniques aim to reduce the computational complexity at each iteration and the number of iterations. In Figure 1 and Figure 2 of Section 4, we compare the accelerated and unaccelerated ALM on synthetic data.

The first technique conducts inner iterations during only one step for $U$ and $V$, which is also adopted in [22]:

$$V_{t+1} = (\mu V I + \beta_t U_t^T U_t)^{-1} \beta_t U_t^T (X - E_t - \frac{Y_t}{\beta_t}), \quad (21)$$

$$(U_{t+1})_{si} = \max(\|Q_{si}\|_2 - \frac{\mu U_i}{\beta_t} \frac{Q_{si}}{\|Q_{si}\|_2}, 0)$$

where $Q_{si} = (U_i - (U_i V_i + E_i - X + \frac{\mu}{\beta_t}) V_{t+1}^T / \xi(V_i))_{si}$. $i = 1, 2, \ldots, K$. Only one $V_{t+1}$ is replaced here to facilitate the later proof. Although we solve (10a) and (U, $V$) at the same time with only one inner iteration, the numerical values show that the Algorithm 2 converges in about ten steps.

The computational complexity primarily stems from the matrix multiplication at each iteration. In the present case, some columns from matrix $U$ are zero owing to the utilization of group norm regularization. This fact inspires the second technique, that is, we delete the zero columns in $U$ and the corresponding rows in $V$ before performing matrix multiplication. In numerical experiments, we find that $r \leq K_{t+1} \leq K_t \leq K$. Here, $K_t$ is the number of non-zero columns of $U$ at the $t$-th iteration. Next, we offer a theoretical proof to ensure that deleting the zero vector column does not affect convergence.

**Theorem 2:** When updating $U$, $V$ by (22) and (21), if the $i$-th column in $U$ is a zero vector column, then this column is always a zero vector in subsequent iterations.

**Proof:** According to (21), we get:

$$\mu V_i V_{t+1} = \beta_t U_t^T (X - E_t - \frac{Y_t}{\beta_t} U_t V_{t+1}).$$

If $(U_{t+1})_{si} = 0$, then $(V_{t+1})_{si} = 0$. Based on (22), we have $Q_{si} = 0$, so $(U_{t+1})_{si} = 0$.

Therefore, the second technique does not affect the convergence, and it speeds up the calculation. We provide a detailed comparison of ALM and AALM in terms of synthetic data in the next section. By applying the above acceleration techniques, we arrive at Algorithm 2 as below.

D. TIME COMPLEXITY

The time complexity for AALM algorithms depends on two factors: the total number of iterations and the computational complexity of each iteration. The numerical results show that our algorithm converges rapidly and only needs about ten iterations. So, we focus on discussing the computational cost per iteration.

From Algorithm 2, we see that the computational complexity arises from matrix multiplication. At the $t$-th iteration, the computational complexity of AALM is $O(K_t^2 n + K_t^2 m + K_t m n)$. Here, $K_t$ is the number of non-zero vector columns in $U$ at the $t$-th iteration. Since $r \leq K_{t+1} \leq K_t \leq K$, the time complexity per iteration for AALM is $O(m n)$.

E. SUBSPACE SEGMENTATION (CLUSTERING)

Similar to Liu et al. [23], we design the following algorithm to perform subspace segmentation (clustering) based on $U^*, V^*$ obtained by solving (7).

**Algorithm 2** Accelerated ALM (AALM) for GNRFM

**Input:** Data $X$, an overestimated rank $K$, and hyperparameters $\mu_U$, $\mu_V$.

**Initialize:** $U_0 \in R^{m \times K}$, $V_0 \in R^{K \times n}$ are obtained by SVD of $X$ ($U_0 = \hat{U}$, $V_0 = \hat{S} \hat{V}$), $Z_0 = Y_0 = 0$, $\beta_0, \rho > 1$, $\zeta \in (0, 1)$, $\nu \in (0, 1)$;

**while** not convergent **do**

1. With $Y = Y_t$, and $\beta = \beta_t$, compute $U_{t+1}, V_{t+1}$ according to (21) and (22) to find an approximate solution for (10). Delete zero columns in $U$ and corresponding rows of $V$. Then, update $E_{t+1}$ according to (16);

2. Set $Y_{t+1} = Y_t + \beta_t(U_{t+1} V_{t+1} + E_{t+1} - X)$;

3. If $t > 0$ and $\|U_{t+1} V_{t+1} + E_{t+1} - X\| \leq \zeta \|U_t V_t + E_t - X\|$, then set $\beta_{t+1} = \beta_t$. Otherwise, set

$$\beta_{t+1} = \max\{\nu \beta_t, \|Y_{t+1}\|^{1+\nu}\};$$

4. Set $t \leftarrow t + 1$.

**end while**

**Algorithm 3** Subspace Segmentation (Clustering)

**Input:** Data $X$, an overestimated rank $K$, hyperparameters $\mu_U$, $\mu_V$, and number $k$ of subspaces;

1. Obtain the minimizer $U^*, V^*$ by Algorithm 2;

2. Compute $Z^* = X^* U^* V^*$;

3. Compute the skinny SVD: $Z^* = \hat{U} \hat{S} \hat{V}^T$;

4. Compute the skinny SVD: $Z^* = \hat{U} \hat{S} \hat{V}^T$;

5. Obtain an affinity matrix $W = (w_{ij}) = (||U_i V_j^T||_2^2);$

6. Use $W$ to perform $N$-Cut and segment the data samples into $k$ clusters.
In the third and sixth steps, one SVD decomposition is needed. For small data sets, this does not take too much time. For large-scale data, the Nyström approximation is also a popular family of methods to replace SVD [31], especially for Spectral Clustering [32]. The data set tested in this article is not particularly large, so the calculation of these two parts is not as critical. In summary, Algorithm 3 describes how to use the solution obtained by GNRFM for clustering.

IV. NUMERICAL EXPERIMENTS

In this section, we test the efficiency of our algorithm and compare it with other algorithms. We implement our algorithm on a PC with 3.2GHZ AMD Ryzen 7 2700 Processor and 16GB of running memory. All computations are done in Matlab 2016b and few tasks are written with C++. We compare our algorithm with three methods: LADMAP(A) [20], IRLS [21], and HMFALM [22]. The first method is based on model (4), which is faster than other SVD algorithms because it uses an adaptive adjustment penalty term to accelerate convergence and uses skinny SVD instead of traditional SVD. The first method also reduces the complexity from $O(n^2)$ to $O(rn^2)$, where $r$ is the predicted rank of $Z$. IRLS smooths the objective function by introducing regular terms and then uses the weighted least squares method to solve the variables alternately. Although the singular value decomposition is not required during the algorithm, the matrix product complexity is still $O(n^3)$. During the solution process, the Matlab command `lyap` is used to solve the Sylvester equation (sometimes the solution of the equation is not unique, and the program is terminated), but in some problems, the number of iterations is less than that of LADMPA(A). HMFALM, which is based on the matrix factorization model (6), does not require calculating SVD; so as to be $O(rmn)$ complexity, where $m$ is the dimension of the data. The outer loop finds the rank, which starts from 1 and increases by step $d$. Under each outer loop, the inner loop is calculated iteratively until the stop condition is met to break out of the inner loop, and until the best rank interval is obtained to find the optimal $r$ one by one. HMFALM is faster than the first two algorithms, but it is very sensitive to the hyperparameter $\mu$, and the anti-noise ability is not good without a regularization term.

Our model GNRFM adds the group norm regularization term to the matrix factorization model (6) and uses the nature of the group norm regularization term; the factor matrix has zero columns so as to adaptively reduce the rank. Although our rank starts to decrease from a large number $K$, it only takes a few steps to iterate from a large rank to a small rank. The numerical results show that our AALM algorithm converges in about ten iterations for (7). The stopping criteria in our numerical experiments is defined as follows:

$$||U_i V_i + E_i - X||_F / ||X||_F < \varepsilon,$$

where $\varepsilon$ is a moderately small number. In the following numerical experiments, we adopt the classic evaluation index indicators, Accuracy (Acc) and Normalized Mutual Information (NMI), to measure the clustering results. The larger the Acc and NMI values are, the better the clustering performance is.

A. EXPERIMENTS ON SYNTHETIC DATA

We first compare the ALM and AALM (before and after acceleration) on the synthetic data. For the inner iteration of ALM, we try two stopping criteria:

1. The inner iteration stops in five fixed steps.
2. The stop criterion of the inner iteration converges when

$$|||U_i V_i (i+1) - U_i V_i (i)||_F |||U_i V_i (i)||_F - 1| < \varepsilon.$$

The construction method for noisy synthetic data is the same as in [1], [20], [22], [33]. The specific construction procedure is as follows. First, we denote the number of subspaces as $s$, and the number of bases in each subspace as $r$, while the dimension of the data is $d$. For the first subspace, we construct the basis $B_1$, which is a random orthogonal matrix with the dimension $d \times r$. Basis $B_i | i=2 ... m$ in the corresponding subspace is obtained by $B_{i+1} = TB_i$, where $T$ is a random rotation matrix. This ensures that these subspaces are independent of each other, and the basis in each subspace is linear independent. Then, in the $i$-th space, we use the basis to generate $p$ samples: $X_i = B_i P_i$, where $P_i \in R^{p \times d}$ is independent and identically distributed, obeying the standard normal distribution $N(0,1)$. Then, we randomly contaminate 20% of the data, such as the data vector $x$, by adding noise according to the following formula:

$$x = x + \sigma ||x|| L_2 \times \eta_{d \times 1},$$

where $\eta$ is a zero mean unit variance Gaussian noise vector. Finally, we get the data matrix $X = [X_1, X_2, ..., X_i] \in \mathbb{R}^{d \times sp}$.

We denote $s = 40, p = 50, d = 2000, r = 5$, and $\sigma = 0.05$ generate synthetic data as described above. In Figure 1 and Figure 2, ALM and AALM are compared.
| (s, p, \tilde{d}, \tilde{r}) | Method | Time(s) | Ite | Acc(%) | NMI | Time(s) | Ite | Acc(%) | NMI | Time(s) | Ite | Acc(%) | NMI |
|-----------------|--------|---------|-----|--------|-----|---------|-----|--------|-----|---------|-----|--------|-----|
| (10, 20, 200, 5) | HMFLM  | 0.0536  | 62  | 26.83  | 0.2647 | 0.2027  | 196 | 100.00 | 1.0000 | 0.2757 | 224 | 98.83  | 0.9798 |
|                 | LADMAP(A) | 0.4703  | 48  | 98.00  | 0.9654 | 3.5960  | 326 | 100.00 | 1.0000 | 14.384 | 1209 | 90.00  | 0.8603 |
|                 | AALM   | 0.0370  | 11  | 100.00 | 1.0000 | 0.0333  | 10  | 100.00 | 1.0000 | 0.0310 | 9    | 100.00 | 1.0000 |
| (15, 20, 200, 5) | HMFLM  | 0.1193  | 75  | 46.33  | 0.4981 | 0.4767  | 191 | 100.00 | 1.0000 | 1.1207 | 259 | 83.44  | 0.7793 |
|                 | LADMAP(A) | 0.8577  | 49  | 99.00  | 0.9847 | 6.1437  | 273 | 100.00 | 1.0000 | 29.571 | 1149 | 84.22  | 0.8187 |
|                 | AALM   | 0.0927  | 11  | 100.00 | 1.0000 | 0.0803  | 10  | 100.00 | 1.0000 | 0.0690 | 9    | 100.00 | 1.0000 |
| (20, 25, 500, 5) | HMFLM  | 0.6727  | 105 | 91.93  | 0.9025 | 1.7920  | 180 | 100.00 | 1.0000 | 5.0133 | 282 | 81.33  | 0.7531 |
|                 | LADMAP(A) | 2.7910  | 72  | 100.00 | 1.0000 | 29.445  | 508 | 99.07  | 0.9872 | 120.57 | 1656 | 91.60  | 0.8848 |
|                 | AALM   | 0.3430  | 11  | 100.00 | 1.0000 | 0.2953  | 10  | 100.00 | 1.0000 | 0.2450 | 9    | 100.00 | 1.0000 |
| (30, 30, 900, 5) | HMFLM  | 3.0670  | 123 | 99.82  | 0.9986 | 11.048  | 219 | 99.63  | 0.9952 | 22.075 | 269 | 80.56  | 0.7604 |
|                 | LADMAP(A) | 16.706  | 99  | 99.96  | 0.9995 | 95.704  | 446 | 88.63  | 0.9277 | 292.40 | 900  | 89.33  | 0.8829 |
|                 | AALM   | 1.8723  | 14  | 98.37  | 0.9807 | 1.4730  | 11   | 100.00 | 1.0000 | 1.1347 | 9    | 100.00 | 1.0000 |
| (35, 40, 1400, 5) | HMFLM  | 13.146  | 165 | 100.00 | 1.0000 | 53.117  | 242 | 80.55  | 0.7572 | 53.511 | 242  | 80.55  | 0.7589 |
|                 | LADMAP(A) | 97.378  | 182 | 86.62  | 0.9374 | 263.81  | 401 | 96.67  | 0.9562 | 1877.1 | 1884 | 90.14  | 0.8744 |
|                 | AALM   | 6.3650  | 16  | 98.88  | 0.9854 | 5.0650  | 13   | 99.98  | 0.9997 | 3.7963 | 10   | 100.00 | 1.0000 |
| (40, 50, 2000, 5) | HMFLM  | 33.242  | 142 | 99.98  | 0.9998 | 122.37  | 230 | 80.53  | 0.7541 | 123.54 | 230  | 80.62  | 0.7572 |
|                 | LADMAP(A) | 477.31  | 344 | 91.43  | 0.9167 | 404.64  | 266 | 93.73  | 0.9181 | 7957.4 | 3330 | 88.17  | 0.8487 |
|                 | AALM   | 16.472  | 17  | 98.67  | 0.9832 | 13.770  | 14   | 99.42  | 0.9928 | 10.161 | 10   | 100.00 | 1.0000 |

In Figure 1, the horizontal axis represents time(s), and the vertical axis is obtained after log_10 transformation of error ||U_i V_j + E_0 - X||_F/||X||_F. In Figure 2, the vertical axis represents the relative error of E_0, which is noise added into the synthetic data. Figure 1 compares the convergence of ALM and AALM, and Figure 2 compares the results of ALM and AALM algorithms to capture noise. The green line represents the inner iteration with five fixed steps. The red line illustrates the inner iteration with a single fixed step, but the zero vector columns are not deleted. The blue line is the inner iteration with one step and deletes the zero vector columns of each U. From Figure 1 and Figure 2, we see that when AALM uses the two acceleration techniques, it converges fastest and obtains the best recovery result. When ALM converges within each inner iteration, it requires thefewest outer iteration steps (11 steps), but it is the slowest. Comparing the blue line with the red line, we observe that deleting the zero vector columns validates our previous analysis; with no effect on the convergence and result, these deletions save memory space and speed up the calculations. From Figures 1 and 2, we see that the inner iteration does not need to converge; even if one step is adopted, this greatly reduces the calculation time.

From Table 2 to Table 4, we use LADMAP(A), HMFLM, and AALM separately to obtain the corresponding affinity matrix on synthetic data with different levels of noise, and then Algorithm 3 is adopted to perform clustering. The goal is to verify noise resistance and sensitivity to hyperparameters between the AALM and several other algorithms for comparison. If the noise level is too high, then the information from the data is lost. So, for the intensity of the noise, we select \( \sigma = [0.05, 0.1, 0.2] \), that is 5%, 10%, 20% of the original data. For the selection of hyperparameters, we select \( \mu = [0.1, 0.2, 0.5] \) for the three algorithms LADMAP(A), IRLS, HMFLM (the parameters mentioned by these three articles). Parameter \( \mu \) for LADMAP(A) and IRLS is the regularized parameter for resisting noise, From model (4), it is apparent that the larger the \( \mu \), the better the anti-noise effect. For the HMFLM model, parameter \( \mu \) plays a role in finding ranks. With respect to our algorithm, \( (\mu_U = 1, \mu_V = 10) \), \( (\mu_U = 1, \mu_V = 20) \), and \( (\mu_U = 1, \mu_V = 50) \) are selected. \( \mu_U \) and \( \mu_V \) are various regularized parameters for resisting noise, and \( \mu_U \) leads the column-sparse factor matrix U to adaptively find ranks. The larger the values of \( \mu_U \) and \( \mu_V \), the better the noise resistance. The greater the value of \( \mu_U \), the faster the rank drop. For the other parameters for IRLS and LADMAP(A) algorithms, we select optimal parameters.
TABLE 3. Numerical results on synthetic data ($\sigma = 0.1$).

| $(s, p, \bar{d}, \bar{r})$ | Method | Time(s) | It | Acc(%) | NMI | Time(s) | It | Acc(%) | NMI | Time(s) | It | Acc(%) | NMI |
|---------------------------|---------|---------|---|--------|-----|---------|---|--------|-----|---------|---|--------|-----|
| (10,20,200,5)            | HMFALM  | 0.0346  | 46 | 0.67  | 0.1716 | 0.1597 | 166 | 99.93 | 0.9971 | 0.4837 | 294 | 83.00 | 0.7937 |
|                          | LADMAP(A) | 0.3403 | 48 | 96.50 | 0.9399 | 3.0467 | 283 | 99.33 | 0.9885 | 10.223 | 728 | 84.67 | 0.8042 |
|                          | AALM  | 0.0460 | 11 | 99.33 | 0.9884 | 0.0387 | 10 | 100.00 | 1.0000 | 0.0320 | 9 | 99.67 | 0.9942 |
| (15,20,200,5)            | HMFALM  | 0.0700 | 40 | 13.44 | 0.1552 | 0.4567 | 182 | 99.78 | 0.9967 | 1.1640 | 263 | 81.00 | 0.7436 |
|                          | LADMAP(A) | 0.7403 | 47 | 94.85 | 0.9776 | 5.1337 | 229 | 99.67 | 0.9957 | 36.790 | 1028 | 83.00 | 0.8153 |
|                          | AALM  | 0.1087 | 10 | 100.00 | 1.0000 | 0.0967 | 9 | 100.00 | 1.0000 | 0.0817 | 9 | 99.78 | 0.9967 |
| (20,25,500,5)            | HMFALM  | 0.5487 | 89 | 40.93 | 0.4913 | 2.7977 | 213 | 96.20 | 0.9654 | 7.4137 | 268 | 81.00 | 0.7515 |
|                          | LADMAP(A) | 0.3330 | 71 | 99.93 | 0.9991 | 32.993 | 553 | 70.40 | 0.8203 | 151.55 | 1664 | 82.07 | 0.7622 |
|                          | AALM  | 0.4077 | 11 | 99.27 | 0.9897 | 0.3210 | 9 | 100.00 | 1.0000 | 0.2960 | 9 | 100.00 | 1.0000 |
| (30,30,900,5)            | HMFALM  | 2.4840 | 97 | 93.89 | 0.9254 | 21.109 | 256 | 80.48 | 0.7607 | 21.159 | 256 | 80.59 | 0.7651 |
|                          | LADMAP(A) | 25.521 | 113 | 93.96 | 0.9672 | 130.25 | 509 | 89.11 | 0.8645 | 824.31 | 2127 | 84.04 | 0.8034 |
|                          | AALM  | 2.1263 | 14 | 97.00 | 0.9622 | 1.5400 | 10 | 99.89 | 0.9986 | 1.3823 | 9 | 100.00 | 1.0000 |
| (35,40,1400,5)           | HMFALM  | 16.596 | 141 | 98.76 | 0.9843 | 50.914 | 228 | 80.57 | 0.7582 | 51.199 | 228 | 80.67 | 0.7593 |
|                          | LADMAP(A) | 201.92 | 70 | 65.14 | 0.7546 | 181.78 | 2023 | 84.79 | 0.8120 | 1304.00 | 1233 | 85.43 | 0.8155 |
|                          | AALM  | 6.9800 | 17 | 99.57 | 0.9945 | 5.9900 | 14 | 99.74 | 0.9965 | 3.9077 | 9 | 100.00 | 1.0000 |
| (40,50,2000,5)           | HMFALM  | 116.01 | 216 | 80.50 | 0.7557 | 117.16 | 216 | 80.65 | 0.7555 | 118.17 | 216 | 80.43 | 0.7542 |
|                          | LADMAP(A) | 600.83 | 391 | 89.95 | 0.8780 | 4628.1 | 2151 | 82.98 | 0.7851 | 3577.4 | 1549 | 87.18 | 0.8350 |
|                          | AALM  | 16.258 | 17 | 99.60 | 0.9947 | 14.914 | 15 | 99.62 | 0.9949 | 11.301 | 11 | 100.00 | 1.0000 |

Table 4 to Table 4 feature synthetic data with different degrees of noise and different dimensions, and almost all of our AALM algorithm results perform the best in terms of clustering and high speed. For Table 2, which features a low-level noise situation ($\sigma = 0.05$), the most accurate of the three algorithms is more than 90% accurate for different dimensional problems, but the HMFALM and LADMAP(A) algorithms are very sensitive to parameters; that is, the results change greatly if parameters change a little, and there is no consistent parameter $\mu$ for all synthetic data. For Table 3, which features moderate noise ($\sigma = 0.1$), our AALM algorithm results are still the best, with a speed increase of more than six times that of HMFALM and about forty times greater than LADMAP(A). Particularly in a high-dimensional situation $d = 2000$, our AALM is more accurate than the other two algorithms by more than 10%. For Table 4, which has the highest noise ($\sigma = 0.2$), our clustering accuracy and NMI are significantly higher than those of the other two algorithms. As a result, although our model (7) has one more hyperparameter than (6), our model is not sensitive to hyperparameters, while the clustering result of the other two models is greatly

based on the corresponding article, and we select $\epsilon = 10^{-5}$, $\beta_1 = 1$, $\beta_{\max} = 10^5$, and $\rho = 2$ for the HMFALM algorithm with the searching gap $d = 0.025 \times n$. We select $\epsilon = 10^{-5}$, $\beta_1 = 1$, $\beta_{\max} = 10^5$, and $\rho = 2$ as the other parameters in our algorithm. $\epsilon$ is the parameter for stopping criteria, and $\beta$ is an increased penalty parameter for the AALM algorithm. For all results, we run the algorithm three times, and take the average as the result for each synthetic data point. We illustrate the best results for each case in bold font.

Table 2 to Table 4 feature synthetic data with different degrees of noise and different dimensions, and almost all of our AALM algorithm results perform the best in terms of clustering and high speed. For Table 2, which features a low-level noise situation ($\sigma = 0.05$), the most accurate of the three algorithms is more than 90% accurate for different dimensional problems, but the HMFALM and LADMAP(A) algorithms are very sensitive to parameters; that is, the results change greatly if parameters change a little, and there is no consistent parameter $\mu$ for all synthetic data. For Table 3, which features moderate noise ($\sigma = 0.1$), our AALM algorithm results are still the best, with a speed increase of more than six times that of HMFALM and about forty times greater than LADMAP(A). Particularly in a high-dimensional situation $d = 2000$, our AALM is more accurate than the other two algorithms by more than 10%. For Table 4, which has the highest noise ($\sigma = 0.2$), our clustering accuracy and NMI are significantly higher than those of the other two algorithms. As a result, although our model (7) has one more hyperparameter than (6), our model is not sensitive to hyperparameters, while the clustering result of the other two models is greatly
affected by the hyperparameter $\mu$. In addition, when $\sigma = [0.05, 0.1, 0.2]$, our clustering accuracy is the best, and even in some cases it is upwards of 20% more accurate than other algorithms. So, the GNRFM model, when introduced with the group norm regularization term, has good noise immunity and is robust.

Specifically, we take $(s, p, \tilde{d}, \tilde{r} = 10, 20, 200, 5)$ and $(\mu_U = 1, \mu_V = 50)$ from Table 2 and draw Figure 3 and Figure 4 for verification. At this time, the data come from ten subspaces; each space has 20 vectors, and each vector is 20 dimensions. Figure 3 shows the values of an affinity matrix, which is obtained by Algorithm 3 after the AALM algorithm solves the GNRFM model with color images. It can be seen that the affinity matrix is clearly divided into ten blocks, and the correlation between different subspaces is very small, and the internal correlation in the subspace is very high, the affinity matrix is well portrayed and accurately segments the subspace. In Figure 4, we use t-SNE [34] to visualize our affinity results. From Figure 4, we see that the affinity results are clearly divided into ten categories, so our models and algorithms are effective for subspace segmentation.

### B. EXPERIMENTS ON REAL DATA

In this section, we test the clustering effectiveness of our algorithm in sport motion segmentation datasets (the Hopkins155 dataset [35] and HARUS dataset [36]) and face segmentation datasets (the Extended Yale B dataset [37] and CMU PIE dataset [38]). Description of the four data sets are shown in Table 5.

The Hopkins155 dataset contains 156 data sequences; each video data sequence contains from 39 to 550 data vectors (from two or three motion modes), and the dimension in each data vector is 72 (24 frames $\times$ 3). We specify the number of classes (two or three classes) in each data sequence, and take advantage of HMFALM, LADM, IRLS, and AALM respectively in these 156 sequences to solve the affinity matrix and conduct clustering. In Table 4, we give the total accuracy, average NMI, average iteration steps, and average time for the data series in the conditions of two motions, three motions, and all motions. Among them, HMFALM, LADM, and IRLS, we select the value of $\mu = 2.4$ (the optimal parameters tested by the authors in their article). With respect to the AALM algorithm, we select $\mu_U = 0.005$, $\mu_V = 3$.

As seen from Table 6, our algorithm is faster than the other three algorithms. For the NMI index with three motions, the results of our algorithm are not as good as IRLS, but our algorithm is much faster than IRLS. For video data, instant clustering is very important. Overall our models and algorithms have achieved the best results.
TABLE 6. Comparison of motion segmentation using different algorithms on the Hopkins155 dataset.

| Problem  | Two Motions | Three Motions | All Motions |
|----------|-------------|---------------|-------------|
|          | Time (s)    | Iter. (s)     | Acc(%)      | Time (s) | Iter. (s) | Acc(%) | Time (s) | Iter. (s) | Acc(%) | NMI |
| HMFALM   | 0.0481      | 124           | 96.34       | 0.8794   | 0.0726    | 137    | 95.06    | 0.8902   | 0.0398  | 127 | 95.99 | 0.8819 |
| LADMAP(A) | 74.064      | 28724         | 96.63       | 0.8884   | 120.20    | 37564  | 95.71    | 0.9103   | 84.711  | 30764 | 96.21 | 0.8935 |
| IRLS     | 33.819      | 189           | 97.21       | 0.9053   | 63.148    | 182    | 95.90    | 0.9140   | 40.5873 | 188  | 96.81 | 0.9073 |
| AALM     | 0.0224      | 14            | 97.63       | 0.8979   | 0.0285    | 14     | 95.17    | 0.8970   | 0.0238  | 14   | 96.87 | 0.8977 |

TABLE 7. Comparison of face clustering with different algorithms on the HARUS dataset.

| HARUS    | Time | Iter. | Acc(%) | NMI |
|----------|------|-------|--------|-----|
| HMFALM   | 81.170 | 222  | 55.44  | 0.5497 |
| LADMAP(A) | 15706 | 7220  | 59.27  | 0.5220 |
| IRLS     | N.A   | N.A   | N.A    | N.A  |
| AALM     | 1.3908 | 5     | 80.34  | 0.6683 |

FIGURE 5. Visualization of the affinity results of HARUS using t-SNE.

The Human Activity Recognition Using Smartphones Dataset (HARUS) contains sensor signals data collected by sensors with a group of 30 volunteers carrying out 6 activities. The HARUS dataset contains 10,299 signals, and each signal is a 561-dimensional feature. In Table 7, we illustrate the accuracy, NMI, iteration steps, and time for four algorithms used on the HARUS dataset. For HMFALM, LADMAP(A), and IRLS, we select μ = 0.01 (the optimal parameters tested by [33], because a large μ value is too time-consuming), with respect to the AALM algorithm; we also select μ_U = 0.005, μ_V = 3. Because the data dimensions are too large, instead of using Algorithm 3 to convert Z into an affinity matrix (SVD requires too much time), we use formula \( W = (|Z| + |Z^T|) / 2 \) to conduct the transformation.

From Table 7, we see that for the large data set, HARUS, the IRLS algorithm does not work. In the remaining algorithms, our AALM algorithm takes less than two seconds, and it is almost 20% more accurate than the other two algorithms. Then, we use t-SNE to visualize the affinity results in Figure 5; we see that the data is roughly separated into six categories with a little error.

In Figure 6, we show the Extended Yale B dataset and CMU PIE dataset. The Extended Yale B dataset contains 38 subjects (people), and each subject has 64 facial images. The CMU PIE dataset contains 68 subjects (people), and each subject has 170 facial images. The Extended Yale B dataset also contains noise from different angles of light. Figure 6 above shows 20 pictures from one person's face where data has lighting noise such that some faces cannot be seen clearly or even become dark. For instance, the image in the fourth picture cannot even be identified by the human eye. The CMU PIE dataset features human expressions in addition to light noise, so clustering is more difficult. Similar to Lu et al. [21], we conduct two experiments in each dataset by constructing the first five subjects and the first ten subjects into a dataset X. First, we resize all the pictures to 32 × 32. Second, to reduce noise, we project them to a 30-dimensional (80) subspace for five subjects and a 60-dimensional (160) subspace for ten subjects by principle component analysis (PCA) for the Extended Yale B (CMU PIE) dataset. Third, by applying HMFALM, LADM, IRLS, and AALM to solve the low-rank representation problem, we obtain different affinity matrices. At last, we compare the clustering results using Algorithm 3 with different affinity matrices. We set parameter \( \mu = 1.5 \) for HMFALM, LADM, IRLS, and our algorithm; \( \mu_U = 1, \mu_V = 20 \) for AALM.

As can be seen from Table 8, the result of the IRLS algorithm is the worst. The remaining three algorithms achieve the same accuracy and NMI, but our algorithm is the fastest. We use t-SNE to visualize the affinity results of the Extended Yale B dataset. From the five subjects in Figure 7, we see that the data is roughly separated into five categories with a little error. From Table 9, which illustrates the CMU PIE dataset, it is apparent that our algorithm achieves the best results. For the large-scale problem of ten subjects in Table 9, our algorithm is much faster than the others. Our algorithm is fast and effective for face clustering.

In summary, our algorithm achieves the best accuracy with the fastest computing speed in real problems with sports motion data (Hopkins155 dataset and HARUS dataset) and facial data (Extended Yale B dataset and CMU PIE dataset). In addition, our AALM algorithm can immediately provide the affinity results for data.
V. CONCLUSION

In this paper, we first propose the use of the group norm regularized factorization model (GNRFM) to solve the problem of low-rank representation with minimal rank, and then we apply it to subspace segmentation. The calculation of the traditional nuclear norm approximation method is very complicated as $O(n^3)$, while the calculation of our group norm model is $O(rnm)$. Compared with the traditional factorization model, which greedily searches for ranks, we adaptively find the rank by introducing the group norm regularization term, which greatly reduces the number of iteration steps. In addition, the group norm regularization term also boasts anti-noise effects and makes the model more robust. On synthetic data and real data, our GNRFM model and the algorithm designed for it achieve excellent clustering results. Furthermore, our AALM algorithm only requires about ten iterations, so the speed is much faster than traditional algorithms; thus, our algorithm plays a role in the process of instant rapid clustering in the era of big data. In the future, we will also consider applying the affinity matrix obtained by the GNRFM to AP clustering [39], Multi-view affinity learning [40], [41], and spectral clustering with interplay manner [41]. In particular, our group norm regularized factorization method can be used for a series of low-rank problems. Some theoretical proofs given by authors will be provided in the future.

REFERENCES

[1] G. Liu, Z. Lin, and Y. Yu. “Robust subspace segmentation by low-rank representation,” in Proc. Int. Conf. Mach. Learn., vol. 1, 2010, pp. 663–670.
K.-C. Toh and S. Yun, “An accelerated proximal gradient algorithm for large-scale l2,1-regularized linear least squares problems,” *SIAM J. Optim.*, vol. 22, no. 4, pp. 1364–1397, Aug. 2012.

Z. Lin, M. Chen, and Y. Ma, “The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices,” *L. Zhang*, *X. Wang*, and *W. Jia*, “Active constraint spectral clustering based on hessian matrix,” *Soft Comput.*, vol. 24, no. 3, pp. 2318–2390, Feb. 2020.

S. Ding, L. Cong, Q. Hu, H. Jia, and Z. Shi, “A multilayer p-spectral clustering algorithm,” *Knowledge-Based Syst.*, vol. 164, pp. 371–377, Jun. 2019.

K.-C. Toh and S. Yun, “An accelerated proximal gradient algorithm for linear programming,” *Pacific J. Optim.*, vol. 6, no. 3, pp. 615–640, Nov. 2010.

Z. Lin, M. Chen, and Y. Ma, “The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices,” *J. Struct. Biol.*, vol. 181, no. 2, pp. 116–127, Feb. 2013.

Z. Lin, R. Liu, and Z. Su, “Linearized alternating direction method with adaptive penalty for low-rank representation,” *Sep. 2011*, arXiv:1109.0367. Accessed: Jun. 5, 2020. [Online]. Available: http://arxiv.org/abs/1109.0367

C. Lu, Z. Lin, and S. Yan, “Smoothed low rank and sparse matrix recovery by iteratively reweighted least squares minimization,” *IEEE Trans. Image Process.*, vol. 24, no. 2, pp. 646–654, Feb. 2015.

B. Chen, Z. Yang, and Z. Yang, “An algorithm for low-rank matrix factorization and its applications,” *Neurocomputing*, vol. 275, pp. 1012–1020, Jun. 2018.

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.*, vol. 35, no. 1, pp. 171–184, Jan. 2013.

W. Siming and L. Zhourchen, “Analysis and improvement of low rank representation for subspace segmentation,” Jul. 2011, arXiv:1107.1561. Accessed: Jun. 5, 2020. [Online]. Available: http://arxiv.org/abs/1107.1561

J. P. Costeira and T. Kanade, “A multibody factorization method for independently moving objects,” *Int. J. Comput. Vis.*, vol. 29, no. 3, pp. 159–179, 1998.

D. L. Sun and C. Fevotte, “Alternating direction method of multipliers for non-negative matrix factorization with the beta-divergence,” in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Florence, Italy, May 2014, pp. 6203–6205.

Y. Shen, Z. Wen, and Y. Zhang, “Augmented lagrangian alternating direction method for matrix separation based on low-rank factorization,” *Optim. Methods Softw.*, vol. 29, no. 2, pp. 239–263, Mar. 2014.

Y. Xu, W. Yin, Z. Wen, and Y. Zhang, “An alternating direction algorithm for matrix completion with nonnegative factors,” *Frontiers Math. China*, vol. 7, no. 2, pp. 365–384, Apr. 2012.

Z. Lu and Y. Zhang, “An augmented lagrangian approach for sparse principal component analysis,” *Jul. 2009*, arXiv:0907.2079. Accessed: Jun. 5, 2020. [Online]. Available: http://arxiv.org/abs/0907.2079

S. Boyd, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2010.

M. Li, W. Bi, J. T. Kwok, and B.-L. Lu, “Large-scale Nystrom kernel matrix approximation using randomized SVD,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 1, pp. 152–164, Jan. 2015.

L. He, N. Ray, Y. Guan, and H. Zhang, “Fast large-scale spectral clustering via explicit feature mapping,” *IEEE Trans. Cybern.*, vol. 49, no. 3, pp. 1058–1071, Mar. 2019.

S. Xiao, W. Li, D. Xu, and D. Tao, “FaLaRR: A fast low rank representation solver,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Boston, MA, USA, Jun. 2015, pp. 4612–4620.

L. van der Maaten and G. Hinton, “Visualizing data using t-SNE,” *J. Mach. Learn. Res.*, vol. 9, no. 2605, pp. 2579–2605, Nov. 2008.

R. Tron and R. Vidal, “A benchmark for the comparison of 3-D motion segmentation algorithms,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Minneapolis, MN, USA, Jun. 2007, pp. 1–8.

D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyesortiz, “Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine,” in *Ambient Assisted Living and Home Care*, vol. 7657, J. Bravo, R. Hervás, and M. Rodríguez, Eds. Berlin, Germany: Springer, 2012, pp. 216–223.

A. S. Georgiades, 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.*, vol. 23, no. 6, pp. 643–660, Jun. 2001.

T. Sim, S. Baker, and M. Bsat, “The CMU pose, illumination, and expression (PIE) database,” in *Proc. 5th IEEE Int. Conf. Autom. Face Gesture Recognit.*, Washington, DC, USA, May 2002, pp. 53–58.

B. J. Frey and D. Dueck, “Clustering by passing messages between data points,” *Science*, vol. 315, no. 5814, pp. 972–976, Feb. 2007.

C.-D. Wang, J.-H. Lai, C. Y. Suen, and J.-Y. Zhu, “Multi-exemplar affinity propagation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 9, pp. 2223–2237, Sep. 2013.

B.-Y. Liu, L. Huang, C.-D. Wang, S. Fan, and P. S. Yu, “Adaptively weighted multiview proximity learning for clustering,” *IEEE Trans. Cybern.*, early access, Dec. 16, 2020, doi: 10.1109/TCYB.2019.2955388.

XISHUN WANG (Member, IEEE) received the B.S. degree in mathematics and the M.E. degree in financial engineering from Soochow University, Suzhou, China, in 2014 and 2017, respectively, where he is currently pursuing the Ph.D. degree with the Center for Financial Engineering.

His current research interests include machine learning and low-rank analysis and financial engineering.

IEEE Access
ZHOUWANG YANG received the bachelor’s, master’s, and Ph.D. degrees in mathematics from the University of Science and Technology of China (USTC), in 1997, 2000, and 2005, respectively.

He worked at Seoul National University as a Postdoctoral Researcher, from 2006 to 2007. He was a Visiting Scholar with the School of Industrial and Systems Engineering (ISyE), Georgia Institute of Technology, from 2010 to 2011. He is currently a Professor with the School of Mathematical Sciences, USTC. He has been working on computation and optimization in data science. His main research interests include geometric modeling and processing, data-driven optimization modeling, mathematical theory of machine learning, and so on.

XINGYE YUE received the B.S. degree in mathematical education, the M.S. degree in applied mathematics, and the Ph.D. degree in applied mathematics from Soochow University, Suzhou, Jiangsu, China, in 1988, 1991, and 1998, respectively.

From 1991 to 2006, he was a Lecturer, an Associate Professor, and a Professor with the School of Mathematical Sciences, Soochow University. From 2006 to 2009, he was a Professor with the Department of Mathematics, University of Sciences and Technology of China. Since 2009, he has been a Professor with the Center for Finance Engineering, Soochow University. His research interests include mathematical modeling and simulation of multiscale problems and numerical partial differential equations with applications on finance, biology, and material sciences. He is the author of more than 40 articles on applied and computational mathematics.

HUI WANG received the B.S. degree in information and computing Science and the M.S. degree in computational mathematics from Soochow University, Suzhou, Jiangsu, China, in 2013 and 2016, respectively.

Since 2016, she has been a Data Analysis Engineer with the State Grid Corporation Customer Service Center. Her research interests include data mining of power data, recognition of power customer appeals, and mathematical modeling.