Large-Scale Subspace Clustering via k-Factorization

Jicong Fan
fanjicong@cuhk.edu.cn
The Chinese University of Hong Kong (Shenzhen) and Shenzhen Research Institute of Big Data
Shenzhen, China

ABSTRACT

Subspace clustering (SC) aims to cluster data lying in a union of low-dimensional subspaces. Usually, SC learns an affinity matrix and then performs spectral clustering. Both steps suffer from high time and space complexity, which leads to difficulty in clustering large datasets. This paper presents a method called k-Factorization Subspace Clustering (k-FSC) for large-scale subspace clustering. K-FSC directly factorizes the data into k groups via pursuing structured sparsity in the matrix factorization model. Thus, k-FSC avoids learning affinity matrix and performing eigenvalue decomposition, and has low (linear) time and space complexity on large datasets. This paper proves the effectiveness of the k-FSC model theoretically. An efficient algorithm with convergence guarantee is proposed to solve the optimization of k-FSC. In addition, k-FSC is able to handle sparse noise, outliers, and missing data, which are pervasive in real applications. This paper also provides online extension and out-of-sample extension for k-FSC to handle streaming data and cluster arbitrarily large datasets. Extensive experiments on large-scale real datasets show that k-FSC and its extensions outperform state-of-the-art methods of subspace clustering.

CCS CONCEPTS

• **Computing methodologies** → **Cluster analysis**; • **Information systems** → **Clustering**.

KEYWORDS

Large-scale clustering; Subspace clustering; Spectral clustering; Matrix factorization

ACM Reference Format:
Jicong Fan. 2021. Large-Scale Subspace Clustering via k-Factorization. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467267

1 INTRODUCTION

Subspace clustering [35, 38, 42, 43] assumes that the data points lie in a union of low-dimensional subspaces and segments the data into different groups corresponding to different subspaces. Classical subspace clustering algorithms such as sparse subspace clustering (SSC) [15], low-rank representation (LRR) [29], and their variants [19, 26, 30, 50] are based on the self-expressive property [15]. These algorithms[1] have two major steps. First, they learn an affinity matrix, of which the time complexity is $O(n^2)$ or even $O(n^3)$ in every iteration of the optimization. The second step is to perform spectral clustering on the affinity matrix, of which the eigenvalue decomposition of the Laplacian matrix has polynomial time complexity. As a result, these algorithms are not applicable to large-scale data [7] owing to their high time and space complexity.

Recently, a few fast subspace or spectral clustering algorithms were proposed for large-scale datasets [8, 10, 18, 20, 27, 28, 36, 44, 45, 48, 49]. For instance, [8] proposed a landmark-based spectral clustering algorithm: 1) produces a few landmark points using k-means; 2) computes the features of all data points via exploiting eigenvectors of the affinity matrix obtained from the landmark points; 3) performs k-means on the features to get the clusters. In [36], the authors treated large-scale subspace clustering as an out-of-sample extension problem of SSC on a few selected landmark data points, in which the clustering problem on the remainders is solved via classification. In [9], the authors proposed an algorithm to directly optimize the normalized cut model and used an anchor-based strategy to extend the algorithm to large-scale data. In [32], a method called S^5C was proposed as a scalable variant of SSC. The method first selects a small subset of the data points by performing sparse representation iteratively; then it performs sparse representation again for all data points using the selected samples and constructs an affinity matrix for all data points; lastly, it uses orthogonal iteration to compute the required eigenvectors for clustering.

Although the aforementioned large-scale subspace clustering methods have achieved considerable success in numerous applications, they still have a few limitations. First, those methods often start with a few samples of the dataset and then expand the representation coefficients to all data. Thus the clustering cannot effectively exploit the whole information of the dataset, which may reduce the clustering accuracy. Second, those methods usually have to store the affinity matrix and compute eigenvectors, which prevent the application to extremely large datasets. Finally, those methods are not effective in handling sparse noise, outliers, missing data, and streaming data, which are pervasive in real applications.

To handle the aforementioned problems, this paper presents a method called k-Factorization Subspace Clustering (k-FSC), which directly factorizes the data matrix into k groups corresponding to k subspaces. The contributions of this work are as follows.

(1) The paper proposes a group-sparse factorization model for subspace clustering. The method k-FSC does not need to learn an

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467267

[1] This paper focuses on a relatively narrow definition of and approach to subspace clustering. More general problems and methods of subspace clustering can be found in [38].

affinity matrix and perform spectral clustering. The time and space complexity of the method are linear with the number of data points.

(2) The paper provides theoretical guarantees for the effectiveness of the k-FSC model.

(3) The paper provides an efficient algorithm with convergence guarantee for the nonconvex nonsmooth optimization of k-FSC.

(4) The paper provides online extension and out-of-sample extension for k-FSC to handle arbitrarily large datasets.

(5) The paper extends k-FSC to robust clustering that is able to handle sparse noise, outliers, and missing data.

Extensive experiments show that the proposed methods[2] outperform the state-of-the-art methods of large-scale clustering.

The remainder of this paper is structured as follows. Section 2 elaborates the proposed k-FSC method. Section 3 is the optimization. Section 4 provides a few extensions of k-FSC. Section 5 discusses the connection with previous work. Section 6 details the experiments. Section 7 presents the conclusion of this paper.

2 K-FACTORIZATION SUBSPACE CLUSTERING

Throughout the paper, we use the following notations. $\boldsymbol{x}$: column vector. $\boldsymbol{X}$: matrix. $[k]$: $\{1, 2, \ldots, k\}$. $\boldsymbol{X}_{:j}$: column-j of $\boldsymbol{X}$. $\boldsymbol{X}_j$: matrix with index j. $[\boldsymbol{X}, \boldsymbol{Y}]$: column-wise stack. $[\boldsymbol{X}; \boldsymbol{Y}]$: row-wise stack. $\boldsymbol{X}^{(j)}$: index-j sub-matrix of $\boldsymbol{X}$. $\|\cdot\|$: Euclidean norm of vector. $\|\cdot\|_2$: spectral norm of matrix. $\|\cdot\|_F$: Frobenius norm of matrix. $\|\cdot\|_1$: ℓ_1 norm of vector or matrix. $|\cdot|$: absolute value of scalar, vector, or matrix. $\mathbb{1}(f)$: 1 if f is true; 0 if f is false.

We first give the following assumption.

Assumption 1. *The columns of data matrix $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ are drawn from a union of k low-dimensional subspaces: $\boldsymbol{x}_i = \boldsymbol{U}^{(j)}\boldsymbol{z}_i$ if $\boldsymbol{x}_i \in \mathcal{S}_j$, where $\boldsymbol{U}^{(j)} \in \mathbb{R}^{m\times d_j}$ are the orthogonal bases of $\mathcal{S}_j$, $j \in [k]$, and $i \in [n]$. The number of data points lying in $\mathcal{S}_j$ is n_j and $d_j < \min\{m, n_j\}$.*

Our goal is to perform subspace clustering on $\boldsymbol{X}$ given by Assumption 1. In contrast to conventional subspace clustering methods, we in this paper propose to directly factorize $\boldsymbol{X}$ into k groups corresponding to k subspaces. Intuitively, we want to solve

$$\underset{\boldsymbol{P},\boldsymbol{U},\boldsymbol{Z}}{\text{minimize}} \ \|\boldsymbol{X}\boldsymbol{P} - \boldsymbol{U}\boldsymbol{Z}\|_F^2, \tag{1}$$

where $\boldsymbol{P} \in \mathbb{R}^{n\times n}$ is a permutation matrix, $\boldsymbol{U} = [\boldsymbol{U}^{(1)}, \ldots, \boldsymbol{U}^{(k)}] \in \mathbb{R}^{m\times\sum_{j=1}^k d_j}$ are the subspace bases, $\boldsymbol{U}^{(j)} \in \mathbb{R}^{m\times d_j}$, and $\boldsymbol{U}^{(j)\top}\boldsymbol{U}^{(j)} = \boldsymbol{I}_{d_j}$, for $j \in [k]$. $\boldsymbol{Z}$ is a block diagonal matrix, i.e.

$$\boldsymbol{Z} = \begin{bmatrix} \boldsymbol{Z}^{(1)} & \ldots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \ldots & \boldsymbol{Z}^{(k)} \end{bmatrix} \in \mathbb{R}^{\sum_{j=1}^d d_j\times n},$$

where $\boldsymbol{Z}^{(j)} \in \mathbb{R}^{d_j\times n_j}$, for $j \in [k]$. The minimum of the objective function in (1) is 0. In fact, it is difficult to solve (1) directly because of the presence of $\boldsymbol{P}$.

Notice that in (1) we can replace $\boldsymbol{U}$ with $\boldsymbol{D} \in \mathbb{S}_D$, where

$$\mathbb{S}_D := \{[\boldsymbol{D}^{(1)}, \ldots, \boldsymbol{D}^{(k)}] \in \mathbb{R}^{m\times\sum_{j=1}^k d_j} : \boldsymbol{D}^{(j)} \in \mathbb{R}^{m\times d_j}, \|\boldsymbol{D}_{:i}^{(j)}\| \le 1, \forall\, i \in [d_j], j \in [k]\}.$$

Meanwhile, we replace $\boldsymbol{X}\boldsymbol{P}$ with $\boldsymbol{X}$ and let $\boldsymbol{C} = \boldsymbol{Z}\boldsymbol{P}^{-1} \in \mathbb{S}_C$ where

$$\mathbb{S}_C := \{[\boldsymbol{C}^{(1)}; \ldots; \boldsymbol{C}^{(k)}] \in \mathbb{R}^{\sum_{j=1}^d d_j\times n} : \boldsymbol{C}^{(j)} \in \mathbb{R}^{d_j\times n}, \sum_{i=1}^n \mathbb{1}(\boldsymbol{C}_{:i}^{(j)} \neq \boldsymbol{0}) = n_j, \forall\, j \in [k]; \sum_{j=1}^k \mathbb{1}(\boldsymbol{C}_{:i}^{(j)} \neq \boldsymbol{0}) = 1, \forall\, i \in [n]\}.$$

Namely, $\boldsymbol{C}$ is a sparse matrix, $\boldsymbol{C}^{(j)}$ has n_j nonzero columns, and the number of nonzero groups in each column of $\boldsymbol{C}$ is 1. Thus we see that we actually don't need to determine $\boldsymbol{P}$ explicitly. Instead, we merge $\boldsymbol{P}$ into $\boldsymbol{C}$, which yields the following problem

$$\underset{\boldsymbol{D}\in\mathbb{S}_D,\boldsymbol{C}\in\mathbb{S}_C}{\text{minimize}} \ \|\boldsymbol{X} - \boldsymbol{D}\boldsymbol{C}\|_F^2. \tag{2}$$

Once $\boldsymbol{C}$ is obtained from (2), the clusters can be identified as

$$\boldsymbol{x}_i \in c_j, \quad \text{if } \boldsymbol{C}_{:i}^{(j)} \neq \boldsymbol{0}, \ i \in [n], \tag{3}$$

where c_j corresponds to $\mathcal{S}_j$, $j \in [k]$. The bases of $\mathcal{S}_1, \ldots, \mathcal{S}_k$ can be computed by applying singular value decomposition to the dictionaries $\boldsymbol{D}^{(1)}, \ldots, \boldsymbol{D}^{(k)}$. We call (2) k-Factorization Subspace Clustering (k-FSC). The general idea of k-FSC is shown in Figure 1.

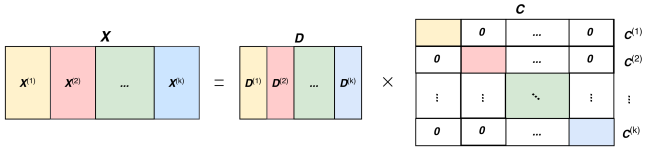

Figure 1: Model structure of k-FSC (we let $\boldsymbol{X} = [\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(k)}]$ for better visualization, though in practice $\boldsymbol{X}$ has been permuted before clustering and $\boldsymbol{C}$ is not block-diagonal; but there always exists a permutation making $\boldsymbol{C}$ block-diagonal).

In (2), the constraints on $\boldsymbol{C}$ make the problem difficult to solve. Particularly, in $\mathbb{S}_C$, the first constraint on $\boldsymbol{C}$ may never be ensured because we usually don't know the number of data points in each clusters. However, later, we will show that, without the first constraint on $\boldsymbol{C}$, we can still cluster the data correctly. To tackle the second constraint on $\boldsymbol{C}$, we propose and expect to minimize the number of nonzero groups in each column of $\boldsymbol{C}$ to 1 via solving

$$\begin{aligned} &\underset{\boldsymbol{D}\in\mathbb{S}_D,\boldsymbol{C}}{\text{minimize}} \ \sum_{i=1}^n \|[\boldsymbol{C}_{:i}^{(1)}, \ldots, \boldsymbol{C}_{:i}^{(k)}]\|_{2,0} = \sum_{j=1}^k \|\boldsymbol{C}^{(j)}\|_{2,0}, \\ &\text{subject to } \ \boldsymbol{D}\boldsymbol{C} = \boldsymbol{X}, \end{aligned} \tag{4}$$

where $\|\cdot\|_{2,0}$ denotes the number of nonzero columns of matrix. The following proposition (we defer all proof of this paper to the appendices) verifies the effectiveness of (4).

Proposition 1. *Suppose $\boldsymbol{X}$ is given by Assumption 1, where $d_1 = \cdots d_k = d$ and $\min_{j\in[d],i\in[n]} |z_{ji}| > 0$. In (4), let $\boldsymbol{D}^{(j)} \in \mathbb{R}^{m\times\hat{d}}$, $\forall j \in [k]$, and let $\boldsymbol{C}^*$ be an optimal solution.*
(a) If the subspaces are independent, i.e. $\dim(\mathcal{S}_1 \cup \mathcal{S}_1 \cdots \cup \mathcal{S}_k) = \sum_{j=1}^k \dim(\mathcal{S}_j) = kd$, and $d \le \hat{d} < 2d$, applying (3) to $\boldsymbol{C}^*$ clusters the data correctly.
(b) If the subspaces share $\bar{d}$ bases ($\bar{d} < d$), i.e. $\dim(\mathcal{S}_1 \cup \mathcal{S}_1 \cdots \cup \mathcal{S}_k) =$

[2]The MATLAB codes of the proposed methods are available at https://github.com/jicongfan/K-Factorization-Subspace-Clustering.

$\sum_{j=1}^{k}(\dim(\mathcal{S}_j)-\bar{d})+\bar{d} = kd-(k-1)\bar{d}$, *and* $d \le \hat{d} < 2d-\bar{d}$, *applying* (3) *to* C^* *clusters the data correctly.*

Nevertheless, it is NP-hard to solve (4) because of the presence of the $\ell_{2,0}$ norm. We define

$$\mathbb{S}_D^+ := \{\boldsymbol{D} : \boldsymbol{D} \in \mathbb{S}_D; \|\boldsymbol{D}^{(l)^\dagger}\boldsymbol{D}^{(j)}\|_2 < 1, \forall\ (l,j) \in [k]\times[k], l \neq j\},$$

where $\boldsymbol{D}^{(l)^\dagger}$ denotes the Moore–Penrose inverse (detailed in Appendix C) of $\hat{\boldsymbol{D}}^{(l)}$. We can solve the following tractable problem

$$\underset{\boldsymbol{D}\in\mathbb{S}_D^+,\boldsymbol{C}}{\text{minimize}} \ \sum_{j=1}^{k}\|\boldsymbol{C}^{(j)}\|_{2,1}, \quad \text{subject to } \ \boldsymbol{D}\boldsymbol{C}=\boldsymbol{X}, \tag{5}$$

where $\|\cdot\|_{2,1}$ denotes the $\ell_{2,1}$ norm [12] of matrix defined by $\|Y\|_{2,1} = \sum_j \|Y_{:j}\|$. It is a convex relaxation of the $\ell_{2,0}$ norm and has been used in many problems such as feature selection [33] and matrix recovery [17]. We have the following theoretical guarantee.

Theorem 1. *Suppose* $\boldsymbol{X}$ *is given by Assumption 1, where* $d_1 = \cdots d_k = d$ *and* $\min_{j\in[d],i\in[n]}|z_{ji}| > 0$. *In* (5), *let* $\boldsymbol{D}^{(j)} \in \mathbb{R}^{m\times\hat{d}}$, $\forall j \in [k]$ *and* $d \le \hat{d} < \infty$. *Let* $\boldsymbol{C}^*$ *be an optimal solution. Then applying* (3) *to* $\boldsymbol{C}^*$ *clusters the data correctly.*

Since real data are often noisy, we relax (5) to

$$\underset{\boldsymbol{D}\in\mathbb{S}_D,\boldsymbol{C}}{\text{minimize}} \ \frac{1}{2}\|\boldsymbol{X}-\boldsymbol{D}\boldsymbol{C}\|_F^2 + \lambda\sum_{j=1}^{k}\|\boldsymbol{C}^{(j)}\|_{2,1}, \tag{6}$$

where λ is a hyper-parameter to be determined in advance. Note that here we used $\mathbb{S}_D$ instead of $\mathbb{S}_D^+$ because the former is easier to optimize[3]. In experiments, we observed that when using $\mathbb{S}_D$, the constraint with $\mathbb{S}_D^+$ is often fulfilled implicitly provided that the angles between pair-wise subspaces are not too small. The following theorem is able to provide a rule of thumb to set λ.

Theorem 2. *Suppose* $\{\boldsymbol{C}_*, \boldsymbol{D}_*\}$ *is a solution of* (6). *For* $i \in [n]$, *let* $\pi_{i1} = argmax_{1\le j\le k}\|\boldsymbol{D}_*^{(j)^\top}\boldsymbol{x}_i\|$ *and* $\pi_{i2} = argmax_{1\le j\neq\pi_{i1}\le k}\|\boldsymbol{D}_*^{(j)^\top}\boldsymbol{x}_i\|$. *If* $\max_{1\le i\le n}\|\boldsymbol{D}_*^{(\pi_{i2})^\top}\boldsymbol{x}_i\| < \lambda \le \min_{1\le i\le n}\|\boldsymbol{D}_*^{(\pi_{i1})^\top}\boldsymbol{x}_i\|$, *then*

$$\sum_{j=1}^{k}\mathbb{1}(\boldsymbol{C}_{*:i}^{(j)} \neq \boldsymbol{0}) = 1, \forall\ i \in [n].$$

According to Theorem 2, if we have a good initialization of $\boldsymbol{D}$, denoted by $\boldsymbol{D}_0$ (detailed in Section 3.1), we can determine λ as

$$\lambda = \left(\max_{1\le i\le n}\|\boldsymbol{D}_0^{(\pi_{i2})^\top}\boldsymbol{x}_i\| + \min_{1\le i\le n}\|\boldsymbol{D}_0^{(\pi_{i1})^\top}\boldsymbol{x}_i\|\right)/2. \tag{7}$$

Owning to noise and local minima, it is possible that $\sum_{j=1}^{k}\mathbb{1}(\boldsymbol{C}_{:i}^{(j)} \neq \boldsymbol{0}) > 1$ for some $i \in [n]$ when we estimate $\boldsymbol{D}$ and $\boldsymbol{C}$ by (6). Thus, we cannot use (3) to assign the data into clusters. We propose to assign $\boldsymbol{x}_i$ to cluster j if the reconstruction error given by $\boldsymbol{D}^{(j)}$ is the least:

$$\boldsymbol{x}_i \in c_j, \quad j = \text{argmin}_j\ \|\boldsymbol{x}_i - \boldsymbol{D}^{(j)}\hat{\boldsymbol{C}}_{:i}^{(j)}\|^2,\ i \in [n], \tag{8}$$

where $\hat{\boldsymbol{C}}^{(j)} = (\boldsymbol{D}^{(j)^\top}\boldsymbol{D}^{(j)} + \lambda'\boldsymbol{I})^{-1}\boldsymbol{D}^{(j)^\top}\boldsymbol{X}$ and λ' is a small constant e.g. 10^{-5}.

[3] As $\|\boldsymbol{D}^{(l)^\dagger}\boldsymbol{D}^{(j)}\|_2 \le \frac{\|\boldsymbol{D}^{(l)^\top}\boldsymbol{D}^{(j)}\|_2}{\sigma^2_{+\min}(\boldsymbol{D}^{(l)})} \le \frac{\|\boldsymbol{D}^{(l)^\top}\boldsymbol{D}^{(j)}\|_F}{\sigma^2_{+\min}(\boldsymbol{D}^{(l)})}$, one may maximize the smallest nonzero singular value of $\boldsymbol{D}^{(l)}$ and minimize $\|\boldsymbol{D}^{(l)^\top}\boldsymbol{D}^{(j)}\|_2$ or $\|\boldsymbol{D}^{(l)^\top}\boldsymbol{D}^{(j)}\|_F$.

In practice, it is difficult to know $d_1, \ldots, d_k$ beforehand. We set $d_1 = \ldots = d_k = d$, where d is a relatively large number, though it can be arbitrarily large according to Theorem 1. Figure 3 in Section 6.1 and Figure 6 in the Appendix A will show that k-FSC is not sensitive to d and indeed d can be arbitrarily large. Comparing these results with Proposition 1, we see that (5) and (6) are much more flexible than (4) in terms of determining d though they are the relaxed formulations of (4).

3 OPTIMIZATION FOR K-FSC

Problem (6) is nonconvex and nonsmooth. When $\boldsymbol{D}$ (or $\boldsymbol{C}$) is fixed, problem (6) regarding of $\boldsymbol{C}$ (or $\boldsymbol{D}$) is convex. Hence we update $\boldsymbol{D}$ and $\boldsymbol{C}$ alternately.

3.1 Initialization

We can initialize $\boldsymbol{D}$ randomly, e.g. draw the entries of $\boldsymbol{D}$ from $\mathcal{N}(0,1)$. Alternatively, we initialize $\boldsymbol{D}$ by k-means, which may improve the convergence of optimization and clustering accuracy. It is worth mentioning that k-means with Euclidean distance measure cannot exploit subspace information and hence does not give us an effective initialization. Instead, we use cosine similarity, $\cos\theta = \frac{\boldsymbol{x}^\top\boldsymbol{y}}{\|\boldsymbol{x}\|\|\boldsymbol{y}\|}$, as a distance measure in k-means. Two data points lying in a same subspace tend to have larger absolute cosine value than lying in different subspaces. Therefore, k-means with cosine "distance" measure is able to provide a better initialization for $\boldsymbol{D}$ than k-means with Euclidean distance measure. The procedures are: 1) perform k-means with cosine "distance" measure on $\boldsymbol{X}$ (or a subset of $\boldsymbol{X}$ when the dataset is too large) to generate cluster centers $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k$; 2) for $j \in [k]$, let $\boldsymbol{D}_0^{(j)}$ consists of the left singular vectors of a matrix formed by the d columns of $\boldsymbol{X}$ (or the subset) closest to $\boldsymbol{c}_j$. Consequently, we initialize $\boldsymbol{C}$ by $\boldsymbol{C}_0 = (\boldsymbol{D}_0^\top\boldsymbol{D}_0 + \hat{\lambda}\boldsymbol{I})^{-1}\boldsymbol{D}_0^\top\boldsymbol{X}$, where $\hat{\lambda}$ is a small constant such as 10^{-5}.

3.2 Update C

At iteration t, we fix $\boldsymbol{D}$ and solve

$$\underset{\boldsymbol{C}}{\text{minimize}} \ \frac{1}{2}\|\boldsymbol{X}-\boldsymbol{D}_{t-1}\boldsymbol{C}\|_F^2 + \lambda\sum_{j=1}^{k}\|\boldsymbol{C}^{(j)}\|_{2,1}. \tag{9}$$

Decomposing (9), for $j \in [k]$, we aim to solve

$$\underset{\boldsymbol{C}^{(j)}}{\text{minimize}} \ \frac{1}{2}\|\boldsymbol{X}-\bar{\boldsymbol{X}}_j-\boldsymbol{D}_{t-1}^{(j)}\boldsymbol{C}^{(j)}\|_F^2 + \lambda\|\boldsymbol{C}^{(j)}\|_{2,1}, \tag{10}$$

where $\bar{\boldsymbol{X}}_j = \sum_{l\neq j}\boldsymbol{D}_{t-1}^{(l)}\boldsymbol{C}_{t'}^{(l)}$, $t' = t$ if $l < j$, and $t' = t-1$ if $l > j$. Problem (10) has no closed-form solution. Denote

$$\mathcal{L}(\boldsymbol{C}^{(j)}) := \frac{1}{2}\|\boldsymbol{X}-\bar{\boldsymbol{X}}_j-\boldsymbol{D}_{t-1}^{(j)}\boldsymbol{C}^{(j)}\|_F^2.$$

The first order approximation of $\mathcal{L}(\boldsymbol{C}^{(j)})$ at $\boldsymbol{C}_{t-1}^{(j)}$ is

$$\begin{aligned}\hat{\mathcal{L}}(\boldsymbol{C}^{(j)}) :=& \frac{1}{2}\|\boldsymbol{X}-\bar{\boldsymbol{X}}_j-\boldsymbol{D}_{t-1}^{(j)}\boldsymbol{C}_{t-1}^{(j)}\|_F^2 \\ &+ \left\langle \boldsymbol{C}^{(j)}-\boldsymbol{C}_{t-1}^{(j)}, \boldsymbol{G}_{t-1}^{(j)}\right\rangle + \frac{\tau}{2}\|\boldsymbol{C}^{(j)}-\boldsymbol{C}_{t-1}^{(j)}\|_F^2,\end{aligned}$$

where $\boldsymbol{G}_{t-1}^{(j)} = \nabla_{\boldsymbol{C}^{(j)}}\mathcal{L}(\boldsymbol{C}^{(j)}) = -\boldsymbol{D}_{t-1}^{(j)^\top}(\boldsymbol{X}-\bar{\boldsymbol{X}}_j-\boldsymbol{D}_{t-1}^{(j)}\boldsymbol{C}_{t-1}^{(j)})$ and $\tau \ge L_{j,t} := \|\boldsymbol{D}_{t-1}^{(j)}\|_2^2$. As $\hat{\mathcal{L}}(\boldsymbol{C}^{(j)}) \ge \mathcal{L}(\boldsymbol{C}^{(j)})$, we now minimize

$\hat{\mathcal{L}}(\boldsymbol{C}^{(j)}) + \lambda\|\boldsymbol{C}^{(j)}\|_{2,1}$, which is equivalent to

$$\underset{\boldsymbol{C}^{(j)}}{\text{minimize}} \ \frac{\tau}{2}\|\boldsymbol{C}^{(j)} - \boldsymbol{C}_{t-1}^{(j)} + \tau^{-1}\boldsymbol{G}_{t-1}^{(j)}\|_F^2 + \lambda\|\boldsymbol{C}^{(j)}\|_{2,1}. \tag{11}$$

The closed-form solution of (11) is

$$\boldsymbol{C}_t^{(j)} = \Theta_{\lambda/\tau}(\boldsymbol{C}_{t-1}^{(j)} - \tau^{-1}\boldsymbol{G}_{t-1}^{(j)}),$$

where $\Theta_u(\cdot)$ is the column-wise soft-thresholding operator [29]

$$\Theta_u(\boldsymbol{v}) = \begin{cases} \frac{(\|\boldsymbol{v}\|-u)\boldsymbol{v}}{\|\boldsymbol{v}\|}, & \text{if } \|\boldsymbol{v}\| > u; \\ \boldsymbol{0}, & \text{otherwise.} \end{cases} \tag{12}$$

The update strategy for $\boldsymbol{C}$ is actually block coordinate descent in the manner of Gauss-Seidel. We further use extrapolation to accelerate the optimization [47]. The procedures are summarized into Algorithm 1, in which fixing $\eta = 0$ will remove the extrapolation.

Algorithm 1 Update $\boldsymbol{C}$ by Gauss-Seidel method

Input: $\boldsymbol{X}$, $\boldsymbol{D}_{t-1}$, $\boldsymbol{C}_{t-1}$, Δ, $\delta < 1$, $\gamma \geq 1$.
1: **for** $j = 1, 2, \ldots, k$ **do**
2: **if** $t \leq 2$ **then**
3: $\eta_{j,t-1} = 0$.
4: **else**
5: $\eta_{j,t-1} = \delta\sqrt{\frac{\tau_{j,t-2}}{\tau_{j,t-1}}}$.
6: **end if**
7: $\hat{\boldsymbol{C}}_{t-1}^{(j)} = \boldsymbol{C}_{t-1}^{(j)} - \eta_{j,t-1}\Delta^{(j)}$.
8: **end for**
9: $\hat{\boldsymbol{X}} = \boldsymbol{D}_{t-1}\hat{\boldsymbol{C}}_{t-1}$.
10: **for** $j = 1, 2, \ldots, k$ **do**
11: $\boldsymbol{G}_{t-1}^{(j)} = -\boldsymbol{D}_{t-1}^{(j)\top}(\boldsymbol{X} - \hat{\boldsymbol{X}})$.
12: $\tau_{j,t-1} = \gamma\|\boldsymbol{D}_{t-1}^{(j)}\|_2^2$.
13: $\boldsymbol{C}_t^{(j)} = \Theta_{\lambda/\tau_{j,t-1}}(\hat{\boldsymbol{C}}_{t-1}^{(j)} - \boldsymbol{G}_{t-1}^{(j)}/\tau_{j,t-1})$.
14: $\hat{\boldsymbol{X}} = \hat{\boldsymbol{X}} + \boldsymbol{D}_{t-1}^{(j)}(\boldsymbol{C}_t^{(j)} - \hat{\boldsymbol{C}}_{t-1}^{(j)})$.
15: **end for**
16: $\Delta = \boldsymbol{C}_{t-1} - \boldsymbol{C}_t$.
Output: Δ, $\boldsymbol{C}_t$.

In Algorithm 1, we update $\boldsymbol{C}^{(j)}$ sequentially for $j \in [k]$, which is not efficient when k and n are large. To improve the efficiency, we may use Jacobi method to update $\boldsymbol{C}$, which is shown in Algorithm 2 and can be implemented parallelly.

Algorithm 2 Update $\boldsymbol{C}$ by Jacobi method

Input: $\boldsymbol{X}$, $\boldsymbol{D}_{t-1}$, $\boldsymbol{C}_{t-1}$, $\gamma \geq 1$.
1: $\boldsymbol{G} = -\boldsymbol{D}_{t-1}^\top(\boldsymbol{X} - \boldsymbol{D}_{t-1}\boldsymbol{C}_{t-1})$.
2: $\tau = \gamma\|\boldsymbol{D}_t\|_2^2$.
3: **for** $j = 1, 2, \ldots, k$ **do**
4: $\boldsymbol{C}_t^{(j)} = \Theta_{\lambda/\tau}(\boldsymbol{C}_{t-1}^{(j)} - \boldsymbol{G}^{(j)}/\tau)$.
5: **end for**
Output: $\boldsymbol{C}_t$.

3.3 Update D

After $\boldsymbol{C}^{(1)}, \ldots, \boldsymbol{C}^{(k)}$ have been updated, we solve

$$\underset{\boldsymbol{D}\in\mathbb{S}_D}{\text{minimize}} \ \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{D}\boldsymbol{C}_t\|_F^2 \tag{13}$$

by projected gradient descent [34]. Specifically, for $u \in [\vartheta]$,

$$\boldsymbol{D}_{t_u} = \mathcal{P}_\Pi\big(\boldsymbol{D}_{t_{u-1}} - \kappa_t^{-1}(\boldsymbol{X} - \boldsymbol{D}_{t_{u-1}}\boldsymbol{C}_t)(-\boldsymbol{C}_t^\top)\big), \tag{14}$$

where $\kappa_t = \|\boldsymbol{C}_t\boldsymbol{C}_t^\top\|_2$ and $\mathcal{P}_\Pi$ denotes the column-wise projection onto unit ball defined by

$$\mathcal{P}_\Pi(\boldsymbol{v}) = \begin{cases} \boldsymbol{v}, & \text{if } \|\boldsymbol{v}\| \leq 1; \\ \boldsymbol{v}/\|\boldsymbol{v}\|, & \text{otherwise.} \end{cases} \tag{15}$$

Algorithm 3 details the implementation. The following theorem provides the convergence rate of Algorithm 3.

Theorem 3 (Theorem 10.21 in [3]). *Let $\boldsymbol{D}_t^*$ be the optimal solution of* (13). *Denote $\mathcal{L}(\boldsymbol{D}_{t_u}) = \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{D}_{t_u}\boldsymbol{C}_t\|_F^2$, where $\boldsymbol{D}_{t_u} \in \mathbb{S}_D$. Then in Algorithm 3,*

$$\mathcal{L}(\boldsymbol{D}_{t_u}) - \mathcal{L}(\boldsymbol{D}_t^*) \leq \frac{\kappa_t}{2u}\|\boldsymbol{D}_{t_u} - \boldsymbol{D}_t^*\|_F^2. \tag{16}$$

Algorithm 3 Projected gradient method for $\boldsymbol{D}$

Input: $\boldsymbol{X}$, $\boldsymbol{C}_t$, $\boldsymbol{D}_{t-1}$, ϑ,
1: $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{C}_t^\top$, $\boldsymbol{B} = \boldsymbol{C}_t\boldsymbol{C}_t^\top$, and $\kappa_t = \|\boldsymbol{B}\|_2$.
2: $\boldsymbol{D}_{t_0} = \boldsymbol{D}_{t-1}$.
3: **for** $u = 1, 2, \ldots, \vartheta$ **do**
4: $\boldsymbol{G} = -\boldsymbol{A} + \boldsymbol{D}_{t_{u-1}}\boldsymbol{B}$.
5: $\boldsymbol{D}_{t_u} = \mathcal{P}_\Pi(\boldsymbol{D}_{t_{u-1}} - \boldsymbol{G}/\kappa_t)$.
6: **end for**
Output: $\boldsymbol{D}_t = \boldsymbol{D}_{t_\vartheta}$.

In fact there is no need to solve (13) exactly because the problem about $\boldsymbol{C}$ (9) is not exactly solved. We just set a small value (e.g. 5) for ϑ to obtain an inexact $\boldsymbol{D}_t$ and keep the time complexity low.

3.4 The overall algorithm of k-FSC

The entire algorithm of k-FSC is shown in Algorithm 4, in which we have set default values for T, δ, γ, ϑ, and ϵ for convenience. The space complexity is $O(mn + kmd + kdn)$ mainly caused by the storage for $\boldsymbol{X}$, $\boldsymbol{D}$, and $\boldsymbol{C}$. In the update of $\boldsymbol{C}$, the time complexity is $O(kdmn)$ mainly caused by line 9 and k loops of lines 11 and 14 in Algorithm 1. The time complexity of Algorithm 2 is lower than that in Algorithm 1. In the update of $\boldsymbol{D}$, the time complexity is $O(kdmn + \vartheta k^2d^2m)$ mainly contributed by line 1 and ϑ loops of line 4 in Algorithm 3. In many real applications, $n \gg m > d$ holds. Thus by assuming $k^2d^2 \leq n$, the time complexity in each iteration of Algorithm 4 is $O(kdmn + \vartheta mn)$. We see that the time complexity and space complexity of k-FSC are linear with the number of data points n. The time complexity of the k-means and line 9 in the initialization is much lower than that in computing $\boldsymbol{D}$ and $\boldsymbol{C}$.

Figure 2 compares the convergence performance of k-FSC with different solvers for $\boldsymbol{C}$ on a synthetic dataset (see Section 6.1). The Gauss-Seidel method with extrapolation for $\boldsymbol{C}$ provides faster convergence than other methods, though Jacobi method can be implemented parallelly for large dataset with large k. The following theorem provides convergence guarantee for Algorithm 4.

Algorithm 4 k-FSC

Input: $\boldsymbol{X}, k; d, \lambda; T(200), \delta(0.95), \gamma(1), \vartheta(5), \epsilon(10^{-4}); t = 0, \boldsymbol{\Delta} = \boldsymbol{0}$.

1: Normalize the columns of $\boldsymbol{X}$ to have unit ℓ_2 norm
2: Generate $\boldsymbol{D}_0$ randomly or by k-means.
3: $\boldsymbol{C}_0 = (\boldsymbol{D}_0^\top \boldsymbol{D}_0 + \hat{\lambda}\boldsymbol{I})^{-1}\boldsymbol{D}_0^\top \boldsymbol{X}$.
4: **repeat**
5: $\quad t \leftarrow t + 1$.
6: $\quad$ Obtain $\boldsymbol{C}_t$ using Algorithm 1 or Algorithm 2.
7: $\quad$ Obtain $\boldsymbol{D}_t$ using Algorithm 3.
8: **until** $\max\left(\frac{\|\boldsymbol{C}_t - \boldsymbol{C}_{t-1}\|_F}{\|\boldsymbol{C}_{t-1}\|_F}, \frac{\|\boldsymbol{D}_t - \boldsymbol{D}_{t-1}\|_F}{\|\boldsymbol{D}_{t-1}\|_F}\right) \le \epsilon$ or $t = T$
9: Identify the clusters by (3) or (8).

Output: k clusters of $\boldsymbol{X}$.

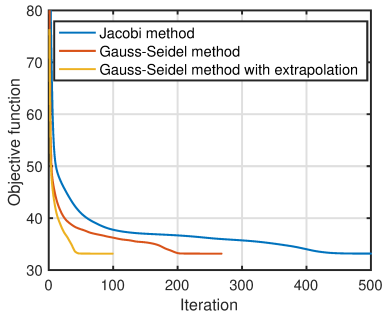

Figure 2: Convergence performance of k-FSC (Algorithm 4) with different solvers for C in clustering a synthetic dataset.

Theorem 4. *Let* $\mathcal{F}(\boldsymbol{C}, \boldsymbol{D}) = \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{D}\boldsymbol{C}\|_F^2 + \lambda \sum_{j=1}^k \|\boldsymbol{C}^{(j)}\|_{2,1}$. *In Algorithm 4 (with Algorithm 1 for* $\boldsymbol{C}$*), for any* $\delta < 1$, $\gamma \ge 1$, *and* $\vartheta \ge 1$, *we have*

$$\lim_{t\to\infty} \|\boldsymbol{C}_{t-1} - \boldsymbol{C}_t\|_F = 0, \quad \lim_{t\to\infty} \|\boldsymbol{D}_{t-1} - \boldsymbol{D}_t\|_F = 0,$$
$$\lim_{t\to\infty} \mathcal{F}(\boldsymbol{C}_{t-1}, \boldsymbol{D}_{t-1}) - \mathcal{F}(\boldsymbol{C}_t, \boldsymbol{D}_t) = 0.$$

4 EXTENSIONS OF K-FSC

4.1 Online and mini-batch optimizations

In many real cases, we observe data points sequentially, which can be well exploited by online learning. In addition, online learning often has low memory cost and low per-iteration time complexity and hence is able to handle very large datasets. K-FSC can be extended to online clustering or solved by mini-batch optimization. Specifically, given a mini-batch data $\boldsymbol{X}_i \in \mathbb{R}^{m\times b}$ at time i, we update $\boldsymbol{D}$ by inexactly solving the following problem

$$\underset{\boldsymbol{D}\in\mathbb{S}_D, \boldsymbol{C}_i}{\text{minimize}} \ \frac{1}{2}\|\boldsymbol{X}_i - \boldsymbol{D}\boldsymbol{C}_i\|_F^2 + \lambda \sum_{j=1}^k \|\boldsymbol{C}_i^{(j)}\|_{2,1}. \tag{17}$$

When $b = 1$, the problem is exactly an online optimization problem. The complete algorithm is shown in Algorithm 5.

4.2 Cluster arbitrarily large dataset

Though the time and space complexity of k-FSC are linear with n, an extremely large n (e.g. $n \ge 10^6$) will still lead to high computational cost. In that case, we propose to perform k-FSC on a few landmark data points generated by k-means and then perform classification. The method is detailed in Algorithm 6. The time complexity per iteration of line 2 (i.e. Algorithm 4) is $O(kdms + \vartheta k^2 d^2 m)$. The time complexity of line 3 and line 4 is $O(kdmn)$.

Algorithm 5 Mini-Batch k-FSC (k-FSC-MB)

Input: Sequential data or randomly partitioned data $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_i \in \mathbb{R}^{m\times b}$, k, d, λ, λ', T_C, T_D.

1: Initialize $\boldsymbol{D}$ randomly or use k-means.
2: **for** $i = 1, 2, \ldots$ **do**
3: $\quad$ Initialize $\boldsymbol{C}_i$
4: $\quad$ Update $\boldsymbol{C}_i$ by performing Algorithm 1 for T_C times.
5: $\quad$ Update $\boldsymbol{D}$ by performing Algorithm 3 with $\vartheta = T_D$.
6: $\quad$ Cluster (similar to lines 8 and 9) if necessary.
7: **end for**
8: Repeat lines 2, 4, 5, 6 for p times if necessary.
9: $\boldsymbol{C}^{(j)} = (\boldsymbol{D}^{(j)\top}\boldsymbol{D}^{(j)} + \lambda'\boldsymbol{I})^{-1}\boldsymbol{D}^{(j)\top}\boldsymbol{X}$, $j \in [k]$.
10: $\boldsymbol{x}_i \in c_j, \quad j = \operatorname{argmin}_j \|\boldsymbol{x}_i - \boldsymbol{D}^{(j)}\boldsymbol{C}_{:i}^{(j)}\|^2$, $i \in [n]$.

Output: k clusters of $\boldsymbol{X}$.

Algorithm 6 k-FSC for arbitrarily large dataset (k-FSC-L)

Input: $\boldsymbol{X} \in \mathbb{R}^{m\times n}$, $s \ll n$, λ' (e.g. 10^{-5}).

1: Let $\boldsymbol{X}_s$ consist of the s centers of k-means on $\boldsymbol{X}$.
2: Run Algorithm 4 on $\boldsymbol{X}_s$ to obtain $\boldsymbol{D}$.
3: $\boldsymbol{C}^{(j)} = (\boldsymbol{D}^{(j)\top}\boldsymbol{D}^{(j)} + \lambda'\boldsymbol{I})^{-1}\boldsymbol{D}^{(j)\top}\boldsymbol{X}$, $j \in [k]$.
4: $\boldsymbol{x}_i \in c_j, \quad j = \operatorname{argmin}_j \|\boldsymbol{x}_i - \boldsymbol{D}^{(j)}\boldsymbol{C}_{:i}^{(j)}\|^2$, $i \in [n]$.

Output: k clusters of $\boldsymbol{X}$.

4.3 Complexity comparison

We analyze the time and space complexity of a few baseline methods. Shown in Table 1, the space complexity of Nyström-ort [20], k-PC [1], k-FSC, and k-FSC-L are much lower than other methods when $n \gg m$. The space complexity of LSC-K [8] and RPCM-F^2 [27] increase quickly when s becomes larger.

For extremely large dataset, in order to achieve high clustering accuracy, we often need a large enough s to exploit sufficient information of the dataset. In k-FSC-L, the complexity is linear with s, which means we may obtain high clustering accuracy by k-FSC-L on extremely large datasets. In contrast, the time complexity of Nyström-ort, LSC-K [8], SSSC [36], RPCM-F^2 [27], S^{5}C [32], and S^{3}COMP-C [10], are at least quadratic with s, which prevents their applications in large-scale clustering demanding high accuracy.

4.4 Sparse noise, outliers, and missing data

In real applications, sparse noise, outliers, and missing data are not uncommon [16, 17]. With slight modification from model (6), k-FSC is able to handle sparse noise, outliers, or/and missing data. For instance, the following model is robust to sparse noise or outliers

$$\underset{\boldsymbol{D}\in\mathbb{S}_D, \boldsymbol{C}}{\text{minimize}} \ \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{D}\boldsymbol{C} - \boldsymbol{E}\|_F^2 + \lambda \sum_{j=1}^k \|\boldsymbol{C}^{(j)}\|_{2,1} + \beta\mathcal{R}(\boldsymbol{E}), \tag{18}$$

Table 1: Time and space complexity ($\rho < 1$: proportion of nonzero entries; k: number of clusters; s: number of selected samples; $b > 1$, $\epsilon < 1$; $\delta < 1$, e.g. 0.8; $\vartheta \geq 1$, e.g. 5).

	Space complexity	Time complexity	
		Iterative	Fixed
k-PC [1]	$O(mn+kmd)$	$O(dmn+kdm^2+m^2n)$	–
SSC [15]	$O(mn+\rho n^2)$	$O(mn^2)$	$O(k\rho n^2)$
Nyström [20]	$O(mn)$	–	$O(msn+s^3)$
LSC-K [8]	$O(mn+sn)$	–	$O(msn+s^2n)$
SSSC [36]	$O(mn+\rho s^2)$	$O(ms^3+k^2s)$	$O(s^2n)$
RPCM-F^2 [27]	$O(mn+s^2+sn)$	$O(ms^2)$	$O(msn+s^2n)$
S^{5}C [32]	$O(mn+\rho n^2)$	–	$O(bms^2+msn+ksn\log\frac{1}{\epsilon})$
S^{3}COMP-C[10]	$O(mn+\rho n^2)$	$O(m\rho n^3(1-\delta))$	$O(k\rho n^2)$
k-FSC	$O(mn+kmd+kdn)$	$O(kdmn+\vartheta mn)$	$O(kdmn)$
k-FSC-MB	$O(mb+kmd+kdb)$	$O(kdmb+\vartheta k^2d^2m)$	$O(kdmn)$
k-FSC-L	$O(ms+kmd+kds)$	$O(kdms+\vartheta k^2d^2m)$	$O(kdmn)$

where $\mathcal{R}(\boldsymbol{E}) = \|\boldsymbol{E}\|_1$ or $\|\boldsymbol{E}\|_{2,1}$. The following model is able to perform clustering and missing data imputation simultaneously.

$$\underset{\boldsymbol{D}\in\mathbb{S}_D,\boldsymbol{C}}{\text{minimize}} \ \frac{1}{2}\|\boldsymbol{M}\odot(\boldsymbol{X}-\boldsymbol{D}\boldsymbol{C})\|_F^2+\lambda\sum_{j=1}^{k}\|\boldsymbol{C}^{(j)}\|_{2,1}, \tag{19}$$

where $\odot$ denotes the Hadamard product and $\boldsymbol{M}$ is a binary matrix with 1 for observed entries and 0 for missing entries. The missing entries can be obtained from $\boldsymbol{D}\boldsymbol{C}$. The optimizations for (18) and (19) can be adapted from Algorithm 4 and will not be detailed here.

5 CONNECTION WITH PREVIOUS WORK

The proposed k-FSC has a connection with nonnegative matrix factorization (NMF) [25]. It is known that k-means clustering can be formulated as NMF [11]. Therefore, both NMF and k-FSC factorize the data into k clusters directly. The difference is that NMF aims to find the cluster centers while k-FSC aims to find the subspaces.

K-FSC is also closely related to the k-plane clustering (k-PC) [4, 23], which aims to minimize the sum of residuals of data points to their assigned subspace. An efficient method to solve k-PC is performing assignment and learn the subspace bases alternately: cluster the data points by their nearest subspaces and update the subspace bases by PCA on the data points in each cluster. K-PC is sensitive to initialization [23], subspace dimension estimation, missing data, and outliers [21].

The model of k-FSC can be regarded as a variant of dictionary learning and sparse coding (DLSC) [31]. Similar to [37, 40, 41][4], k-FSC also considers structured dictionary. It is worth pointing out that, in these previous work, the regularization on the coefficients matrix is ℓ_1 norm. In contrast, k-FSC puts ℓ_{21} norm on the k submatrices of the coefficients matrix to make it be group-sparse, which enables us to factorize the data matrix into k groups directly. In [39], the authors proposed to perform DLSC and clustering alternately, which is-time consuming and not applicable to large datasets.

6 EXPERIMENTS

6.1 Synthetic data

This paper generates[5] synthetic data $\boldsymbol{X} = [\boldsymbol{X}^{(1)},\ldots,\boldsymbol{X}^{(k)}]$ by $\boldsymbol{X}^{(j)} = (\alpha\boldsymbol{A}_0+\boldsymbol{A}^{(j)})\boldsymbol{B}^{(j)}$. Here $\boldsymbol{A}^{(j)}\in\mathbb{R}^{m\times d_0}$ and $\boldsymbol{B}^{(j)}\in\mathbb{R}^{d_0\times n_0}$ are drawn from $\mathcal{N}(0,1)$, $j\in[k]$. $\boldsymbol{A}_0$ is a random matrix drawn from $\mathcal{N}(0,1)$ and α controls the similarity between pair-wise subspaces. We also add random noise to $\boldsymbol{X}$: $\hat{\boldsymbol{X}} = \boldsymbol{X}+\boldsymbol{E}$, where $\boldsymbol{E}$ is drawn from $\mathcal{N}(0,(\beta\sigma_x)^2)$, σ_x denotes the standard deviation of the entries in $\boldsymbol{X}$, and β controls the noise level. We set $k=5$, $m=25$, $d_0=5$, $n_0=50$, and $\alpha=1$.

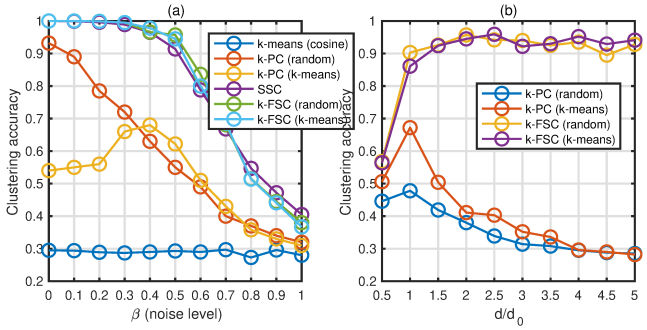

Figure 3: Clustering accuracies of k-means, k-PC, SSC, and k-FSC (with random or k-means initialization): (a) different noise level; (b) k-PC and k-FSC with different estimation (d) of subspace dimension, where $\beta = 0.5$.

Figure 3(a) shows the clustering accuracy (average of 50 trials) of k-means (cosine distance), k-PC [1] with $d = d_0$, SSC [42], and the proposed k-FSC ($d = 2d_0$) in the cases of different noise level. Random initialization and k-means (cosine distance) initialization for k-PC and k-FSC are also compared. We see that k-means failed in all cases though we have used cosine as a distance metric. K-FSC outperformed SSC when the noise level was relatively large; they outperformed k-PC in all cases. Note that in this study, as k-means failed, initialization by k-means provided no significant improvement compared to random initialization. Figure 3 (b) presents the influence of d in k-PC and k-FSC when $\beta = 0.5$. We see that k-PC requires d be equal to the true dimension d_0, otherwise the clustering accuracy decreases quickly when d increases. In contrast, k-FSC is not sensitive to d, even when d is five times of the true dimension of the subspaces. In addition, k-FSC is also not sensitive to λ, which can be found in Appendix A.

To test the clustering performance of k-FSC when the data are corrupted by sparse noise, we use $\hat{\boldsymbol{X}} = [\boldsymbol{X}^{(1)},\ldots,\boldsymbol{X}^{(k)}]+\boldsymbol{E}+\boldsymbol{F}$, where $\boldsymbol{E}$ was defined previously and $\boldsymbol{F}$ is a sparse matrix whose

[4] Structured dictionary was also considered in compressed sensing [14] but the dictionary is not unknown in that case.

[5] All experiments in this paper are conducted in MATLAB on a MacBook Pro with 2.3 GHz Intel Core i5 and 8 GB RAM.

nonzero entries are drawn from $\mathcal{N}(0, \sigma_x^2)$. We let $\beta = 0.1$ and increase the proportion of the nonzero entries (noise density of sparse noise) of $\boldsymbol{F}$ from 0 to 0.5. The clustering accuracy of k-PC, SSC, and k-FSC are reported in Figure 4(a). We see that k-PC is very vulnerable to the sparse noise. Compared to SSC, k-FSC is more robust to the sparse noise and the clustering accuracy is always higher than 0.9 when the noise density is no larger than 0.4.

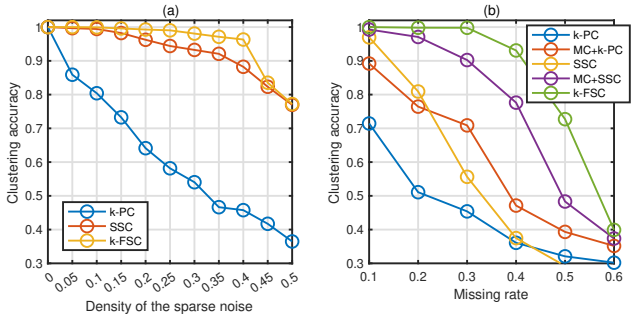

Figure 4: (a) sparse noise; (b) missing data.

We randomly remove a fraction (missing rate) of the entries of the data matrix ($\beta = 0.1$) and test the performance of k-PC, SSC, and k-FSC. In k-PC and SSC, we fill the missing entries with zero. We also use low-rank matrix completion [6] to recover the matrix and then perform k-PC and SSC. The clustering accuracy of 50 repeated trials are reported in Figure 4(b). We see matrix completion has improved the clustering accuracy of k-PC and SSC. Nevertheless, k-FSC has the highest clustering accuracy in all cases. It is worth mentioning that the data matrix is full-rank and hence cannot be well recovered by low-rank matrix completion. That's why the proposed method outperformed MC+k-PC and MC+SSC.

6.2 Real data

We compare k-FSC (Algorithm 4), k-FSC-MB (Algorithm 5), and k-FSC-L (Algorithm 6) with k-means (cosine similarity), k-PC [1], SSC [15], Nyström-orth [20], LSC-K [8], SSSC [36], RPCM-F^2 [27], S^{5}C [32], and S^{3}COMP-C [10]. We use the MATLAB codes shared by their authors. The evaluation are conducted on the following six datasets. **MNIST**: [24] 70,000 gray images (28×28) of handwritten digits. Similar to [10], for each image, we use the scattering convolution network [5] to generate a feature vector of dimension 3472 further reduced to 150 by PCA (use the first 150 right singular vectors of the matrix). **Fashion-MNIST**: [46] 70,000 gray images (28×28) of 10 types of fashion product. The preprocessing is the same as that for MNIST. **Epileptic**: [2] EEG data with 178 features and 11,500 samples in 5 classes. We reduced the feature dimension to 50 by PCA (use the first 50 right singular vectors of the matrix). **Motion Capture Hand Postures**: A UCI [13] dataset with 38 features and 78,095 samples in 5 classes. **Covtype**: A UCI [13] dataset with 54 features and 581,012 samples in 7 classes. **PokerHand**: A UCI [13] dataset with 10 features and 1,000,000 samples in 10 classes. All data are normalized to have unit ℓ_2 norm.

The following parameter settings are used for the six datasets. In k-PC, we set $d = 6, 10, 6, 2, 5, 3$. In Nyström-orth, σ =0.25, 0.5,0.3,0.5,0.2,1 and s = 3000,3000,2000,1500,1500,1000. In LSC-K, $r = 5, 4, 5, 5, 5, 5,$

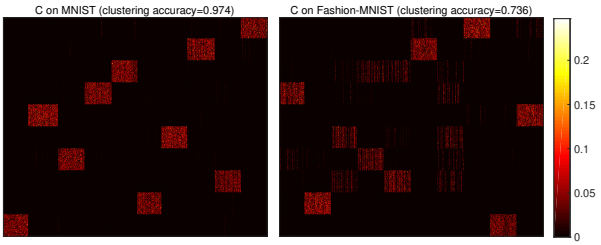

Figure 5: Visualization of $|C|$ given by k-FSC.

Table 2: Performance on MNIST and Fashion-MNIST

		ACC (%)	NMI (%)	Time (s)
MNIST	k-means	95.72±3.38	91.46±3.84	28.7
	k-PC	87.67±6.75	83.21±4.75	41.9
	Nyström	78.56±7.13	76.49±3.80	60.3
	LSC-K	95.83±1.09	90.91±0.78	296.2
	SSSC	82.93±0.39	83.44±0.53	127.7
	NCSC	94.09	86.12	Need GPU
	RPCM-F^2	96.95±0.19	91.87±0.31	54.5
	S^{5}C	94.86±1.37	89.85±1.13	291.6
	S^{3}COMP-C	96.32	/	/
	k-FSC	**97.24**±0.02	**92.58**±0.05	335.9
	k-FSC-MB	97.13±0.04	92.30±0.09	55.2
	k-FSC-L	**97.48**±0.31	**93.45**±0.45	36.9
Fashion-MNIST	k-means	65.51±5.05	65.23±3.03	33.4
	k-PC	61.88±6.31	60.78±4.01	48.8
	Nyström	54.62±3.67	48.33±1.40	60.2
	LSC-K	63.27±2.77	65.60±1.65	290.3
	SSSC	57.90±1.48	60.69±0.88	121.4
	NCSC	72.14	68.60	Need GPU
	RPCM-F^2	65.98±3.19	67.23±1.95	55.8
	S^{5}C	63.13±1.63	66.38±1.34	297.2
	S^{3}COMP-C	59.88±2.19	65.00±0.17	762.6
	k-FSC	**72.73**±3.13	**69.24**±1.94	527.5
	k-FSC-MB	**71.51**±4.08	68.08±3.09	58.5
	k-FSC-L	69.70±4.32	**68.23**±2.45	57.7

and s = 3000, 3000,2000,1500,1000,1000. In SSSC, $\lambda = 10^{-4}$,10^{-2}, 10^{-1},10^{-1}, 10^{-1} and s =3000, 3000, 1500,1500,3500. In RPCM-F^2, $\beta = 0.1, 0.1, 10, 0.5, 0.1$ and s = 3000,3000,1500,1500,1000. In S^{5}C, $\lambda = 0.2, 0.2, 0.3, 0.2$ and s = 3000, 3000, 1500, 1500. In S^{3}COMP-C, $T = 10$, $\beta = 0.8$, and $\lambda = 1, 1, 0.4, 0.5$ for the first four datasets.

In k-FSC, k-FSC-MB, and k-FSC-L, for the six datasets, we set d = 30,30,30,30,20,5, λ = 0.5,0.5,0.5,0.2,0.4,0.1, and $\lambda' = 10^{-5}$. In k-FSC-MB, we set $b = 1000$ and $p = 5, 5, 20, 10, 2, 2$. In k-FSC-L, we set s = 5000, 5000, 1500, 2500, 3500, 5000, namely $s = 500k$ (except Epileptic because it is a relatively small dataset). The numbers of repetitions of k-means in k-PC, LSC-K, k-FSC, and k-FSC-MB are

10 on all datasets. The number of repetitions of k-means in k-FSC-L is 100 on all datasets. Note that according to Theorem 1, d can be arbitrarily large. But in practice we just use a relatively small d (according to the data dimension m) to reduce the computational cost. Since the initialization of $\boldsymbol{D}$ may not be good enough, we still need to tune λ under the guidance of Theorem 2.

Figure 5 shows two examples of $\boldsymbol{C}$ given by k-FSC on MNIST and Fashion-MNIST. We see that k-FSC can find the cluster blocks effectively. The average clustering accuracy (ACC), normalized mutual information (NMI), and time cost[6] of ten repeated trials on MNIST and Fashion-MNIST are reported in Table 2, in which we also compare NCSC [50] (a deep learning method). We see that k-FSC, k-SFC-MB, and k-FSC-L outperformed other methods in terms of ACC and NMI. Meanwhile, k-FSC-MB and k-FSC-L are more efficient than most methods such as S^{5}C and S^{3}COMP-C.

Table 3: Performance on Epileptic and Postures

		ACC (%)	NMI (%)	Time (s)
Epileptic	k-means	23.88±0.09	0.84±0.02	1.3
	k-PC	42.03±1.76	18.12±0.91	2.1
	Nyström	27.21±2.84	4.91±1.55	11.2
	LSC-K	33.75±0.27	14.96±0.23	9.8
	SSSC	38.14±3.27	19.41±2.76	27.1
	RPCM-F^2	38.01±2.43	16.42±1.07	2.3
	S^{5}C	41.42±2.15	22.08±1.79	24.3
	S^{3}COMP-C	41.39±3.68	**26.04**±2.38	436.5
	k-FSC	43.26±2.16	23.82±1.12	21.7
	k-FSC-MB	**43.49**±1.75	24.01±0.98	9.1
	k-FSC-L	**45.40**±0.98	**24.29**±1.33	5.9
Postures	k-means	42.68±2.12	33.61±0.87	7.2
	k-PC	41.41±3.41	21.33±3.65	9.2
	Nyström	43.27±2.78	32.35±0.82	21.6
	LSC-K	46.40±2.44	37.24±1.68	207.7
	SSSC	45.39±3.24	36.71±1.02	20.1
	RPCM-F^2	47.02±2.71	36.41±2.15	23.0
	S^{5}C	46.67±0.41	**38.66**±1.48	451.8
	S^{3}COMP-C	45.26±3.38	36.24±1.49	755.3
	k-FSC	**51.65**±2.26	**39.39**±0.74	173.9
	k-FSC-MB	49.97±2.29	36.15±1.73	24.6
	k-FSC-L	**51.10**±4.73	38.18±2.17	9.8

The results on Epileptic and postures are shown in Table 3. In terms of ACC, the proposed methods outperformed all other methods. In terms of NMI, the proposed methods outperformed all other methods except S^{5}C and S^{3}COMP-C that are time-consuming..

[6]The time cost is the total cost of all procedures. The underlined values are the results reported in the original papers. The '/' means out-of-memory or exceeding 3 hours. On Fashion-MNIST and Postures, S^{3}COMP-C is out of memory. So we perform S^{3}COMP-C on two subsets (20%) of Fashion-MNIST and Postures. The time costs of S^{3}COMP-C can be reduced if performed in parallel.

Table 4: Performance on Covtype and PokerHand

		ACC (%)	NMI (%)	Time (s)
Covtype	k-means	20.84±0.00	3.69±0.00	156.6
	k-PC	37.45±4.16	5.09±0.51	123.7
	Nyström	23.18±0.90	3.75±0.01	635.8
	LSC-K	24.16±1.29	5.73±0.08	4792.5
	SSSC	30.02±1.46	6.48±0.31	332.6
	RPCM-F^2	23.66±0.53	3.75±0.11	2362.2
	S^{5}C	/	/	/
	S^{3}COMP-C	/	/	/
	k-FSC	**43.95**±3.46	5.59±1.64	1762.6
	k-FSC-MB	41.31±3.27	**7.70**±3.76	60.4
	k-FSC-L	**43.72**±2.95	**6.92**±2.77	19.6
PokerHand	k-means	10.47±0.05	0.04±0.00	169.3
	k-PC	12.43±0.42	0.17±0.05	306.5
	Nyström	10.91±0.15	0.08±0.03	995.6
	LSC-K	12.32	0.00	8829.0
	SSSC	19.31	0.20	474.1
	RPCM-F^2	/	/	/
	S^{5}C	/	/	/
	S^{3}COMP-C	/	/	/
	k-FSC	21.82±2.18	**0.33**±0.13	1017.8
	k-FSC-MB	**33.15**±7.09	0.21±0.14	33.2
	k-FSC-L	**22.19**±3.13	**0.39**±0.15	18.6

The results on Covtype and PokerHand are reported in Table 4. These two datasets are more challenging than the previous four datasets because the clusters are highly imbalanced, which will lead to low NMI. On PokerHand, the results of LSC-K and SSSC are from [36]. Since the datasets are too large, S^{5}C and S^{3}COMP-C do not apply. The ACCs of the proposed methods are much higher than other methods. Moreover, the time costs of k-FSC-MB and k-FSC-L are much lower than other methods.

7 CONCLUSION

This paper has presented a linear-complexity method k-FSC for subspace clustering. K-FSC is able to handle arbitrarily large dataset, streaming data, sparse noise, outliers, and missing data. Extensive experiments showed that k-FSC and its extensions are more accurate and efficient than state-of-the-art methods of subspace clustering. This improvement stems from the following aspects. First, k-FSC, k-FSC-MB, and k-FSC-L can utilize much more data points in the learning step while most of the other methods require the subset be small enough to ensure the scalability. Second, in the proposed methods, the number of clusters, as an important information, is directly exploited. Other methods except k-PC do not use the information before the spectral clustering step. K-FSC-MB and k-FSC-L are very efficient in handling very large datasets and are as accurate as k-FSC is. Future study may focus on the sufficient conditions for k-FSC to succeed.

ACKNOWLEDGEMENTS

The work was supported by the research funding T00120210002 of Shenzhen Research Institute of Big Data. The author appreciates the reviewers' valuable time and comments.

REFERENCES

[1] Pankaj K Agarwal and Nabil H Mustafa. 2004. K-means projective clustering. In *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 155–165.

[2] Ralph G Andrzejak, Klaus Lehnertz, Florian Mormann, Christoph Rieke, Peter David, and Christian E Elger. 2001. Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. *Physical Review E* 64, 6 (2001), 061907.

[3] Amir Beck. 2017. *First-order methods in optimization*. SIAM.

[4] Paul S Bradley and Olvi L Mangasarian. 2000. K-plane clustering. *Journal of Global Optimization* 16, 1 (2000), 23–32.

[5] Joan Bruna and Stéphane Mallat. 2013. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence* 35, 8 (2013), 1872–1886.

[6] Emmanuel J. Candès and Benjamin Recht. 2009. Exact Matrix Completion via Convex Optimization. *Foundations of Computational Mathematics* 9, 6 (2009), 717–772. https://doi.org/10.1007/s10208-009-9045-5

[7] Wen-Yen Chen, Yangqiu Song, Hongjie Bai, Chih-Jen Lin, and Edward Y Chang. 2010. Parallel spectral clustering in distributed systems. *IEEE transactions on pattern analysis and machine intelligence* 33, 3 (2010), 568–586.

[8] Xinlei Chen and Deng Cai. 2011. Large scale spectral clustering with landmark-based representation. In *Twenty-fifth AAAI conference on artificial intelligence*. Citeseer.

[9] Xiaojun Chen, Weijun Hong, Feiping Nie, Dan He, Min Yang, and Joshua Zhexue Huang. 2018. Spectral clustering of large-scale data by directly solving normalized cut. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1206–1215.

[10] Ying Chen, Chun-Guang Li, and Chong You. 2020. Stochastic Sparse Subspace Clustering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4155–4164.

[11] Chris Ding, Xiaofeng He, and Horst D Simon. 2005. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proceedings of the 2005 SIAM international conference on data mining*. SIAM, 606–610.

[12] Chris Ding, Ding Zhou, Xiaofeng He, and Hongyuan Zha. 2006. R1-PCA: rotational invariant L1-norm principal component analysis for robust subspace factorization. In *Proceedings of the 23rd international conference on Machine learning*. 281–288.

[13] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[14] Yonina C Eldar, Patrick Kuppinger, and Helmut Bolcskei. 2010. Block-sparse signals: Uncertainty relations and efficient recovery. *IEEE Transactions on Signal Processing* 58, 6 (2010), 3042–3054.

[15] E. Elhamifar and R. Vidal. 2013. Sparse Subspace Clustering: Algorithm, Theory, and Applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 11 (2013), 2765–2781. https://doi.org/10.1109/TPAMI.2013.57

[16] Jicong Fan and Tommy W.S. Chow. 2017. Sparse subspace clustering for data with missing entries and high-rank matrix completion. *Neural Networks* 93 (2017), 36–44.

[17] Jicong Fan, Lijun Ding, Yudong Chen, and Madeleine Udell. 2019. Factor group-sparse regularization for efficient low-rank matrix recovery. In *Advances in Neural Information Processing Systems*. 5104–5114.

[18] Jicong Fan, Zhaoyang Tian, Mingbo Zhao, and Tommy W.S. Chow. 2018. Accelerated low-rank representation for subspace clustering and semi-supervised classification on large-scale data. *Neural Networks* 100 (2018), 39–48.

[19] Jicong Fan, Chengrun Yang, and Madeleine Udell. 2021. Robust Non-Linear Matrix Factorization for Dictionary Learning, Denoising, and Clustering. *IEEE Transactions on Signal Processing* 69 (2021), 1755–1770.

[20] Charless Fowlkes, Serge Belongie, Fan Chung, and Jitendra Malik. 2004. Spectral grouping using the Nystrom method. *IEEE transactions on pattern analysis and machine intelligence* 26, 2 (2004), 214–225.

[21] Andrew Gitlin, Biaoshuai Tao, Laura Balzano, and John Lipor. 2018. Improving K-Subspaces via Coherence Pursuit. *IEEE Journal of Selected Topics in Signal Processing* 12, 6 (2018), 1575–1588.

[22] Markus Haltmeier. 2013. Block-sparse analysis regularization of ill-posed problems via l 2, 1-minimization. In *2013 18th International Conference on Methods & Models in Automation & Robotics (MMAR)*. IEEE, 520–523.

[23] Jun He, Yue Zhang, Jiye Wang, Nan Zeng, and Hanyong Hao. 2016. Robust k-subspaces recovery with combinatorial initialization. In *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 3573–3582.

[24] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[25] Daniel D Lee and H Sebastian Seung. 2001. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*. 556–562.

[26] Chun-Guang Li and Rene Vidal. 2015. Structured sparse subspace clustering: A unified optimization framework. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 277–286.

[27] J. Li, H. Liu, Z. Tao, H. Zhao, and Y. Fu. 2020. Learnable Subspace Clustering. *IEEE Transactions on Neural Networks and Learning Systems* (2020), 1–15.

[28] Jun Li and Handong Zhao. 2017. Large-scale subspace clustering by fast regression coding. In *IJCAI*.

[29] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma. 2013. Robust Recovery of Subspace Structures by Low-Rank Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 1 (2013), 171–184.

[30] Canyi Lu, Jiashi Feng, Zhouchen Lin, Tao Mei, and Shuicheng Yan. 2018. Subspace clustering by block diagonal representation. *IEEE transactions on pattern analysis and machine intelligence* 41, 2 (2018), 487–501.

[31] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. 2009. Online dictionary learning for sparse coding. In *Proceedings of the 26th annual international conference on machine learning*. ACM, 689–696.

[32] Shin Matsushima and Maria Brbic. 2019. Selective sampling-based scalable sparse subspace clustering. In *Advances in Neural Information Processing Systems*. 12416–12425.

[33] Feiping Nie, Heng Huang, Xiao Cai, and Chris H Ding. 2010. Efficient and robust feature selection via joint $\ell_{2,1}$-norms minimization. In *Advances in neural information processing systems*. 1813–1821.

[34] Neal Parikh, Stephen Boyd, et al. 2014. Proximal algorithms. *Foundations and Trends® in Optimization* 1, 3 (2014), 127–239.

[35] Lance Parsons, Ehtesham Haque, and Huan Liu. 2004. Subspace clustering for high dimensional data: a review. *SIGKDD Explor. Newsl.* 6, 1 (2004), 90–105.

[36] Xi Peng, Lei Zhang, and Zhang Yi. 2013. Scalable sparse subspace clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 430–437.

[37] I. Ramirez, P. Sprechmann, and G. Sapiro. 2010. Classification and clustering via dictionary learning with structured incoherence and shared features. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 3501–3508. https://doi.org/10.1109/CVPR.2010.5539964

[38] Kelvin Sim, Vivekanand Gopalkrishnan, Arthur Zimek, and Gao Cong. 2013. A survey on enhanced subspace clustering. *Data mining and knowledge discovery* 26, 2 (2013), 332–397.

[39] Pablo Sprechmann and Guillermo Sapiro. 2010. Dictionary learning and sparse coding for unsupervised clustering. In *2010 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2042–2045.

[40] Yuanming Suo, Minh Dao, Trac Tran, Hojjat Mousavi, Umamahesh Srinivas, and Vishal Monga. 2014. Group structured dirty dictionary learning for classification. In *2014 IEEE International Conference on Image Processing (ICIP)*. IEEE, 150–154.

[41] Zoltán Szabó, Barnabás Póczos, and András Lőrincz. 2011. Online group-structured dictionary learning. In *CVPR 2011*. IEEE, 2865–2872.

[42] R. Vidal. 2011. Subspace Clustering. *IEEE Signal Processing Magazine* 28, 2 (2011), 52–68. https://doi.org/10.1109/MSP.2010.939739

[43] Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and computing* 17, 4 (2007), 395–416.

[44] Shusen Wang, Bojun Tu, Congfu Xu, and Zhihua Zhang. 2014. Exact subspace clustering in linear time. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. 2113–2120.

[45] Lingfei Wu, Pin-Yu Chen, Ian En-Hsu Yen, Fangli Xu, Yinglong Xia, and Charu Aggarwal. 2018. Scalable spectral clustering using random binning features. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2506–2515.

[46] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. arXiv:cs.LG/1708.07747 [cs.LG]

[47] Yangyang Xu and Wotao Yin. 2013. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on imaging sciences* 6, 3 (2013), 1758–1789.

[48] Chong You, Chun-Guang Li, Daniel P Robinson, and René Vidal. 2016. Oracle based active set algorithm for scalable elastic net subspace clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3928–3937.

[49] Chong You, Daniel Robinson, and René Vidal. 2016. Scalable sparse subspace clustering by orthogonal matching pursuit. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3918–3927.

[50] Tong Zhang, Pan Ji, Mehrtash Harandi, Wenbing Huang, and Hongdong Li. 2019. Neural collaborative subspace clustering. *arXiv preprint arXiv:1904.10596* (2019).

A MORE RESULTS ON SYNTHETIC DATA

In Figure 6(a), k-PC does not work when $d/d_0 > 5$ because the subspace dimension is equal or larger than the data dimension($m = 25, d_0 = 5$). K-FSC always has high clustering accuracy even when $d = 50d_0$. Figure 6(b) shows the clustering accuracy of k-FSC ($d = 2d_0$) with different hyper-parameter λ in the cases of different noise level. We see that k-FSC works well with a large range of λ especially when the noise level is low.

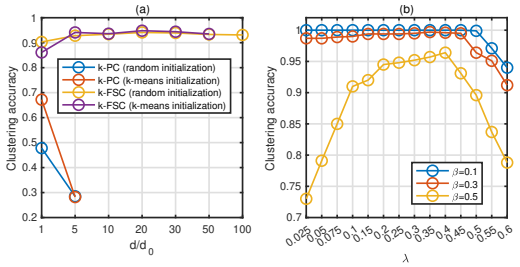

Figure 6: (a) sensitivity of k-PC/k-FSC to d ($\beta = 0.5$); (b) sensitivity of k-FSC to λ in the cases of different noise level β.

B PROOF FOR PROPOSITION 1

Proof. (a) To prove, we need to show that: ① $\sum_{j=1}^{k} \|C^{(j)}\|_{2,0}$ has a minimum under the constraint; ② when the minimum of $\sum_{j=1}^{k} \|C^{(j)}\|_{2,0}$ is attained, all columns of $\boldsymbol{X}$ are correctly clustered.

For ①. Obviously, when all columns of $\boldsymbol{X}$ are correctly clustered according to (3), we have $\sum_{j=1}^{k} \|C^{(j)}\|_{2,0} = n$. If $\sum_{j=1}^{k} \|C^{(j)}\|_{2,0} < n$, $\boldsymbol{C}$ has at least one zero column, which means the corresponding column of $\boldsymbol{X}$ can not be reconstructed and $\boldsymbol{X} \neq \boldsymbol{DC}$. Therefore, under the constraint $\boldsymbol{X} = \boldsymbol{DC}$, we have $\sum_{j=1}^{k} \|C^{(j)}\|_{2,0} \geq n$ and the minimum is attainable.

For ②, we only need to show that when one column of $\boldsymbol{X}$ is not correctly clustered, $\sum_{j=1}^{k} \|C^{(j)}\|_{2,0} \geq n+1$. Without loss of generality, we assume that $\boldsymbol{x}_1 \in \mathcal{S}_j, \boldsymbol{x}_2 \in \mathcal{S}_l$, and $j \neq l$. Suppose that $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are assigned into $\mathcal{S}_p$ corresponding to $\boldsymbol{D}^{(p)}$, where $p \in [k]$. Since the subspaces are independent and $\min_{j\in[d],i\in[n]} |z_{ji}| > 0$, to ensure there exist some $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ such that $\boldsymbol{x}_1 = \boldsymbol{D}^{(p)}\boldsymbol{c}_1$ and $\boldsymbol{x}_2 = \boldsymbol{D}^{(p)}\boldsymbol{c}_2$, the column space of $\boldsymbol{D}^{(p)}$ must contain $\boldsymbol{U}^{(j)}$ and $\boldsymbol{U}^{(l)}$. It indicates $\hat{d} \geq 2d$, which is contradiction to the assumption $\hat{d} < 2d$. Hence, at least one column of $[\boldsymbol{U}^{(j)}, \boldsymbol{U}^{(l)}]$ is contained in the column space of some $\boldsymbol{D}^{(q)}$, where $q \neq l \neq j$. As a result,

$$\boldsymbol{x}_1 = [\boldsymbol{D}^{(p)}, \boldsymbol{D}^{(q)}] \begin{bmatrix} \boldsymbol{c}_1^{(p)} \\ \boldsymbol{c}_1^{(q)} \end{bmatrix} \quad \text{or} \quad \boldsymbol{x}_2 = [\boldsymbol{D}^{(p)}, \boldsymbol{D}^{(q)}] \begin{bmatrix} \boldsymbol{c}_2^{(p)} \\ \boldsymbol{c}_2^{(q)} \end{bmatrix}$$

where $\boldsymbol{c}_1^{(p)}, \boldsymbol{c}_1^{(q)} \neq \boldsymbol{0}$ or $\boldsymbol{c}_2^{(p)}, \boldsymbol{c}_2^{(q)} \neq \boldsymbol{0}$. Therefore, $\sum_{j=1}^{k} \|C^{(j)}\|_{2,0} \geq n+1$, if the data are not correctly clustered. In other words, if $\sum_{j=1}^{k} \|C^{(j)}\|_{2,0} < n+1$, all columns of $\boldsymbol{X}$ are clustered correctly. Together with ①, we finish the proof.

(b) The proof is similar to that for (a) and is omitted for simplicity. □

C PROOF FOR THEOREM 1

Before proving the theorem, we give the following lemmas.

Lemma 1. *Let $\{\hat{C}, \hat{D}\}$ be the optimal solution of* (5)*. Then for all $i \in [n]$, $\sum_{j=1}^{k} \mathbb{1}(\|\hat{C}_{:i}^{(j)}\| \neq \boldsymbol{0}) = 1$.*

Proof. Suppose $\boldsymbol{x} = \sum_{j=1}^{k} \boldsymbol{D}^{(j)}\boldsymbol{c}^{(j)}$ and $\boldsymbol{x} = \boldsymbol{D}^{(l)}\boldsymbol{\alpha}$ where $l \in [k]$. It follows that

$$\boldsymbol{D}^{(l)}\boldsymbol{\alpha} = \sum_{j=1}^{k} \boldsymbol{D}^{(j)}\boldsymbol{c}^{(j)}. \tag{20}$$

The minimum-norm solution of $\boldsymbol{\alpha}$ in (20) is

$$\hat{\boldsymbol{\alpha}} = {\boldsymbol{D}^{(l)}}^{\dagger} \sum_{j=1}^{k} \boldsymbol{D}^{(j)}\boldsymbol{c}^{(j)}, \tag{21}$$

where ${\boldsymbol{D}^{(l)}}^{\dagger} = \boldsymbol{V}\begin{bmatrix} \Sigma^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}\boldsymbol{U}^{\top}$ denotes the Moore–Penrose inverse of $\hat{\boldsymbol{D}}^{(l)}$ and $\boldsymbol{V}, \Sigma, \boldsymbol{U}$ are from the SVD $\boldsymbol{D}^{(l)} = \boldsymbol{U}\begin{bmatrix} \Sigma & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}\boldsymbol{V}^{\top}$. We have

$$\begin{aligned} \|\hat{\boldsymbol{\alpha}}\| =& \|\sum_{j=1}^{k} {\boldsymbol{D}^{(l)}}^{\dagger}\boldsymbol{D}^{(j)}\boldsymbol{c}^{(j)}\| \leq \|\boldsymbol{c}^{(l)}\| + \sum_{j\neq l} \|{\boldsymbol{D}^{(l)}}^{\dagger}\boldsymbol{D}^{(j)}\boldsymbol{c}^{(j)}\| \\ \leq& \|\boldsymbol{c}^{(l)}\| + \sum_{j\neq l} \|{\boldsymbol{D}^{(l)}}^{\dagger}\boldsymbol{D}^{(j)}\|_2 \|\boldsymbol{c}^{(j)}\| \leq \sum_{j=1}^{k} \|\boldsymbol{c}^{(j)}\|, \end{aligned} \tag{22}$$

where the first inequality used the fact ${\boldsymbol{D}^{(l)}}^{\dagger}\boldsymbol{D}^{(l)} = \boldsymbol{I}$ and the third inequality used the condition in $\mathbb{S}_D^+$. In (22), if $\boldsymbol{c}^{(j)} = \boldsymbol{0}$ for all $j \neq l$, the equality holds and then $\hat{\boldsymbol{c}} = \hat{\boldsymbol{\alpha}}$. Because $\boldsymbol{x} = \boldsymbol{D}\hat{\boldsymbol{\alpha}}$, $\hat{\boldsymbol{\alpha}} \neq \boldsymbol{0}$. Now expanding the result to all columns of $\boldsymbol{X}$, we finish the proof. □

Lemma 2. *Suppose $\boldsymbol{x} \in \mathcal{S}_\ell$ and $\boldsymbol{x} = \boldsymbol{D}^{(l)}\boldsymbol{\alpha}$. Denote $\boldsymbol{U}^{(\ell)}$ the basis of $\mathcal{S}_\ell$. The minimum of $\|\boldsymbol{\alpha}\|$ is not attained if some columns of $\boldsymbol{D}^{(l)}$ are not in span$(\boldsymbol{U}^{(\ell)})$.*

Proof. We partition $\boldsymbol{D}^{(l)}$ into two parts $\boldsymbol{D}^{(l)} = [\boldsymbol{D}_\ell^{(l)} \quad \boldsymbol{D}_{-\ell}^{(l)}]$, where $\boldsymbol{D}_\ell^{(l)} \in \mathbb{R}^{m\times d_u}$, $\boldsymbol{D}_{-\ell}^{(l)} \in \mathbb{R}^{m\times(\hat{d}-d_u)}$, and $1 \leq d_u \leq \hat{d}-1$. The columns of $\boldsymbol{D}_{-\ell}^{(l)}$ are not in span$(\mathbf{U}^{(\ell)})$. A smaller $\|\boldsymbol{\alpha}\|$ is obtained when $\boldsymbol{\alpha} \leftarrow [\alpha_1, \ldots, \alpha_{d_u}, 0, \ldots, 0]^{\top}$.

Let $\tilde{\boldsymbol{D}}^{(l)} = [\boldsymbol{D}_\ell^{(l)} \quad \tilde{\boldsymbol{D}}_\ell^{(l)}]$, where $\tilde{\boldsymbol{D}}_\ell^{(j)} \in \mathbb{R}^{m\times(\hat{d}-d_u)}$ and the columns of $\tilde{\boldsymbol{D}}_\ell^{(j)}$ are in span$(\boldsymbol{U}^{(\ell)})$. There is always a $\tilde{\boldsymbol{D}}_\ell^{(l)}$ such that

$$\|\boldsymbol{\alpha}\| > \|\bar{\boldsymbol{\alpha}}\|, \tag{23}$$

where $\tilde{\boldsymbol{D}}^{(l)}\bar{\boldsymbol{\alpha}} = \boldsymbol{D}^{(l)}\boldsymbol{\alpha}$. An example is $\tilde{\boldsymbol{D}}_\ell^{(l)} = [\boldsymbol{\delta}, \ldots, \boldsymbol{\delta}]$, where $\boldsymbol{\delta}$ is the last column of $\boldsymbol{D}_\ell^{(l)}$. Accordingly,

$$\bar{\boldsymbol{\alpha}} = [\alpha_1, \ldots, \alpha_{d_u-1}, \frac{\alpha_{d_u}}{\hat{d}-d_u+1}, \ldots, \frac{\alpha_{d_u}}{\hat{d}-d_u+1}].$$

Obviously, $\|\boldsymbol{\alpha}\| > \sqrt{\sum_{i=1}^{d_u-1} {\alpha_i}^2 + \frac{{\alpha_{d_u}}^2}{\hat{d}-d_u+1}} = \|\bar{\boldsymbol{\alpha}}\|$. □

Now combining Lemma 1 and Lemma 2, we conclude that in the optimal solution of (5), the columns of each $\boldsymbol{D}^{(j)}$ are in the span of one subspace's bases, each column of $\boldsymbol{X}$ is reconstructed by only one sub-matrix of $\boldsymbol{D}$, and cannot be reconstructed by an incorrect sub-matrix (since $\min_{j\in[d],i\in[n]} |z_{ji}| > 0$). This finished the proof.

D PROOF FOR THEOREM 2

Proof. We have the following result.

Lemma 3 ([22]). *The subgradient of $\ell_{2,1}$ norm is*

$$\partial\|\boldsymbol{x}\|_{2,1} = \begin{cases} \boldsymbol{x}/\|\boldsymbol{x}\|, & \text{if } \|\boldsymbol{x}\| > 0; \\ \boldsymbol{z} : \|\boldsymbol{z}\| < 1, & \text{if } \|\boldsymbol{x}\| = 0. \end{cases} \tag{24}$$

The optimality for the problem in the proposition indicates that

$$\boldsymbol{D}^{(j)\top}(\boldsymbol{x} - \boldsymbol{D}\boldsymbol{c}^{(j)}) = \lambda\partial\|\boldsymbol{c}^{(j)}\|_{2,1}, \quad \forall j \in [k].$$

Letting $\boldsymbol{c} = 0$ be the optimal solution, we have

$$\|\boldsymbol{D}^{(j)\top}\boldsymbol{x}\| < \lambda, \quad \forall j \in [k].$$

It means $\lambda > \max_j \|\boldsymbol{D}^{(j)\top}\boldsymbol{x}\|$. Expanding the result for all columns of $\boldsymbol{X}$, we finish the proof. □

E PROOF FOR THEOREM 4

Proof. First, we give the following two lemmas.

Lemma 4 (Lemma 10.4 in [3]). *Denote $\mathcal{L}(\boldsymbol{D}_{t_u}) = \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{D}_{t_u}\boldsymbol{C}_t\|_F^2$, where $\boldsymbol{D}_{t_u} \in \mathbb{S}_D$. Then in Algorithm 3,*

$$\mathcal{L}(\boldsymbol{D}_{t_{u-1}}) - \mathcal{L}(\boldsymbol{D}_{t_u}) \geq \frac{\kappa_t}{2}\|\boldsymbol{D}_{t_{u-1}} - \boldsymbol{D}_{t_u}\|_F^2. \tag{25}$$

Lemma 5 (Lemma 2.1 in [47]). *Let $g(\boldsymbol{u})$ and $h(\boldsymbol{u})$ be two convex functions defined on the convex set $\mathcal{U}$ and $g(\boldsymbol{u})$ be differentiable. Let $f(\boldsymbol{u}) = g(\boldsymbol{v}) + h(\boldsymbol{u})$ and $\boldsymbol{u}^* = \operatorname{argmin}_{\boldsymbol{u}\in\mathcal{U}} \langle\nabla g(\boldsymbol{v}), \boldsymbol{u} - \boldsymbol{v}\rangle + \frac{L}{2}\|\boldsymbol{u} - \boldsymbol{v}\|^2 + h(\boldsymbol{u})$. If*

$$g(\boldsymbol{u}^*) \leq g(\boldsymbol{v}) + \langle\nabla g(\boldsymbol{v}), \mathcal{P}_L(\boldsymbol{v}) - \boldsymbol{v}\rangle + \tfrac{L}{2}\|\boldsymbol{u}^* - \boldsymbol{v}\|^2,$$

then for any $\boldsymbol{u} \in \mathcal{U}$ we have

$$f(\boldsymbol{u}) - f(\boldsymbol{u}^*) \geq \tfrac{L}{2}\|\boldsymbol{u}^* - \boldsymbol{v}\|^2 + L\left\langle \boldsymbol{v} - \boldsymbol{u}, \boldsymbol{u}^* - \boldsymbol{v}\right\rangle.$$

In Section 3.2, we select $\tau_{j,t-1}$ to make

$$\mathcal{L}(\boldsymbol{C}^{(j)}) \leq \mathcal{L}(\hat{\boldsymbol{C}}_{t-1}^{(j)}) + \left\langle \boldsymbol{C}^{(j)} - \hat{\boldsymbol{C}}_{t-1}^{(j)}, \hat{\boldsymbol{G}}^{(j)}\right\rangle + \frac{\tau_{j,t-1}}{2}\|\boldsymbol{C}^{(j)} - \hat{\boldsymbol{C}}_{t-1}^{(j)}\|_F^2.$$

Then use Lemma 5 and let $g = \mathcal{L}$, $h = \lambda\|\cdot\|_{2,1}$, $\boldsymbol{u} = \boldsymbol{C}_{t-1}^{(j)}$, and $\boldsymbol{v} = \hat{\boldsymbol{C}}_{t-1}^{(j)}$. Denote $\mathcal{F}_j(\boldsymbol{C}_t^{(j)}) = \mathcal{L}(\boldsymbol{C}_t^{(j)}) + \lambda\|\boldsymbol{C}_t^{(j)}\|_{2,1}$ and $\mathcal{F}(\boldsymbol{C}_t, \boldsymbol{D}_{t-1}) = \mathcal{L}(\boldsymbol{C}_t, \boldsymbol{D}_{t-1}) + \lambda\sum_{j=1}^k \|\boldsymbol{C}_t^{(j)}\|_{2,1}$. We have

$$\begin{aligned} &\mathcal{F}_j(\boldsymbol{C}_{t-1}^{(j)}) - \mathcal{F}_j(\boldsymbol{C}_t^{(j)}) \\ \geq&\frac{\tau_{j,t-1}}{2}\|\hat{\boldsymbol{C}}_{t-1}^{(j)} - \boldsymbol{C}_t^{(j)}\|_F^2 + \tau_{j,t-1}\left\langle \hat{\boldsymbol{C}}_{t-1}^{(j)} - \boldsymbol{C}_{t-1}^{(j)}, \boldsymbol{C}_t^{(j)} - \hat{\boldsymbol{C}}_{t-1}^{(j)}\right\rangle \\ =&\frac{\tau_{j,t-1}}{2}\|\boldsymbol{C}_{t-1}^{(j)} - \boldsymbol{C}_t^{(j)}\|_F^2 - \frac{\tau_{j,t-1}}{2}\eta_{j,t-1}^2\|\boldsymbol{C}_{t-2}^{(j)} - \boldsymbol{C}_{t-1}^{(j)}\|_F^2 \\ \geq&\frac{\tau_{j,t-1}}{2}\|\boldsymbol{C}_{t-1}^{(j)} - \boldsymbol{C}_t^{(j)}\|_F^2 - \frac{\tau_{j,t-2}}{2}\psi(t-1)\delta^2\|\boldsymbol{C}_{t-2}^{(j)} - \boldsymbol{C}_{t-1}^{(j)}\|_F^2, \end{aligned} \tag{26}$$

where $\psi(t-1) = 0$ if $t \leq 2$ and $\psi(t-1) = 1$ if $t > 2$, according to the setting of $\eta_{j,t-1}$ in Algorithm 4 in the main paper.

It follows that

$$\begin{aligned} &\mathcal{F}(\boldsymbol{C}_{t-1}, \boldsymbol{D}_{t-1}) - \mathcal{F}(\boldsymbol{C}_t, \boldsymbol{D}_{t-1}) = \sum_{j=1}^k \mathcal{F}_j(\boldsymbol{C}_{t-1}^{(j)}) - \mathcal{F}_j(\boldsymbol{C}_t^{(j)}) \\ \geq&\sum_{j=1}^k \Big(\frac{\tau_{j,t-1}}{2}\|\boldsymbol{C}_{t-1}^{(j)} - \boldsymbol{C}_t^{(j)}\|_F^2 - \frac{\tau_{j,t-2}}{2}\psi(t-1)\delta^2\|\boldsymbol{C}_{t-2}^{(j)} - \boldsymbol{C}_{t-1}^{(j)}\|_F^2\Big). \end{aligned} \tag{27}$$

On the other hand, according to Lemma 4, we have

$$\begin{aligned} &\mathcal{F}(\boldsymbol{C}_t, \boldsymbol{D}_{t-1}) - \mathcal{F}(\boldsymbol{C}_t, \boldsymbol{D}_t) = \sum_{u=1}^{\vartheta} \frac{\kappa_t}{2}\|\boldsymbol{D}_{t_{u-1}} - \boldsymbol{D}_{t_u}\|_F^2 \\ =&\frac{\kappa_t}{2}\|\boldsymbol{D}_{t-1} - \boldsymbol{D}_{t_1}\|_F^2 + \frac{\kappa_t}{2}\|\boldsymbol{D}_{t_{\vartheta-1}} - \boldsymbol{D}_t\|_F^2 + \Delta_{D_t} \triangleq \tilde{\Delta}_{D_t}, \end{aligned} \tag{28}$$

where $\Delta_{D_t} = \sum_{u=2}^{\vartheta-1} \frac{\kappa_t}{2}\|\boldsymbol{D}_{t_{u-1}} - \boldsymbol{D}_{t_u}\|_F^2$.

Combining (27) with (28), we have

$$\begin{aligned} &\mathcal{F}(\boldsymbol{C}_{t-1}, \boldsymbol{D}_{t-1}) - \mathcal{F}(\boldsymbol{C}_t, \boldsymbol{D}_t) \geq \sum_{j=1}^k \Big(\frac{\tau_{j,t-1}}{2}\|\boldsymbol{C}_{t-1}^{(j)} - \boldsymbol{C}_t^{(j)}\|_F^2 \\ &- \frac{\tau_{j,t-2}}{2}\psi(t-1)\delta^2\|\boldsymbol{C}_{t-2}^{(j)} - \boldsymbol{C}_{t-1}^{(j)}\|_F^2\Big) + \tilde{\Delta}_{D_t}. \end{aligned} \tag{29}$$

Now summing (29) over t from 1 to T, we arrive at

$$\begin{aligned} &\mathcal{F}(\boldsymbol{C}_0, \boldsymbol{D}_0) - \mathcal{F}(\boldsymbol{C}_T, \boldsymbol{D}_T) \\ \geq&\sum_{t=1}^T\sum_{j=1}^k \Big(\frac{\tau_{j,t-1}}{2}\|\boldsymbol{C}_{t-1}^{(j)} - \boldsymbol{C}_t^{(j)}\|_F^2 \\ &\qquad - \frac{\tau_{j,t-2}}{2}\psi(t-1)\delta^2\|\boldsymbol{C}_{t-2}^{(j)} - \boldsymbol{C}_{t-1}^{(j)}\|_F^2\Big) + \sum_{t=1}^T \tilde{\Delta}_{D_t} \\ =&\sum_{j=1}^k \frac{\tau_{j,0}}{2}\|\boldsymbol{C}_0^{(j)} - \boldsymbol{C}_1^{(j)}\|_F^2 + \sum_{j=1}^k \frac{\tau_{j,T-1}}{2}\|\boldsymbol{C}_{T-1}^{(j)} - \boldsymbol{C}_T^{(j)}\|_F^2 \\ &+ \sum_{t=2}^T\sum_{j=1}^k \frac{(1-\delta^2)\tau_{j,t-1}}{2}\|\boldsymbol{C}_{t-1}^{(j)} - \boldsymbol{C}_t^{(j)}\|_F^2 + \sum_{t=1}^T \tilde{\Delta}_{D_t} \\ \geq&\sum_{j=1}^k \frac{(1-\delta^2)\tau_{j,0}}{2}\|\boldsymbol{C}_0^{(j)} - \boldsymbol{C}_1^{(j)}\|_F^2 \\ &+ \sum_{j=1}^k \frac{(1-\delta^2)\tau_{j,T-1}}{2}\|\boldsymbol{C}_{T-1}^{(j)} - \boldsymbol{C}_T^{(j)}\|_F^2 \\ &+ \sum_{t=2}^T\sum_{j=1}^k \frac{(1-\delta^2)\tau_{j,t-1}}{2}\|\boldsymbol{C}_{t-1}^{(j)} - \boldsymbol{C}_t^{(j)}\|_F^2 + \sum_{t=1}^T \tilde{\Delta}_{D_t} \\ \geq&\sum_{t=1}^T \frac{(1-\delta^2)\bar{\tau}}{2}\|\boldsymbol{C}_{t-1} - \boldsymbol{C}_t\|_F^2 + \sum_{t=1}^T \tilde{\Delta}_{D_t}, \end{aligned} \tag{30}$$

where $\bar{\tau} = \min\{\tau_{j,t} : j = 1, \ldots, k,\ t = 0, \ldots, T\}$. Notice that $\bar{\tau} > 0$ according to its definition. Since $\mathcal{F}(\boldsymbol{C}, \boldsymbol{D})$ is bounded blow and the two parts in the right-hand-side of the last inequality of (30) are nonnegative, letting $T \to \infty$, we have

$$\sum_{t=1}^{\infty} \frac{(1-\delta^2)\bar{\tau}}{2}\|\boldsymbol{C}_{t-1} - \boldsymbol{C}_t\|_F^2 < \infty$$

$$\sum_{t=1}^{\infty} \frac{\kappa_t}{2}\|\boldsymbol{D}_{t-1} - \boldsymbol{D}_{t_1}\|_F^2 + \frac{\kappa_t}{2}\|\boldsymbol{D}_{t_{\vartheta-1}} - \boldsymbol{D}_t\|_F^2 + \Delta_{D_t} < \infty.$$

It follows that

$$\lim_{t\to\infty}\|\boldsymbol{C}_{t-1} - \boldsymbol{C}_t\|_F = 0, \quad \lim_{t\to\infty}\|\boldsymbol{D}_{t-1} - \boldsymbol{D}_t\|_F = 0,$$
$$\lim_{t\to\infty}\mathcal{F}(\boldsymbol{C}_{t-1}, \boldsymbol{D}_{t-1}) - \mathcal{F}(\boldsymbol{C}_t, \boldsymbol{D}_t) = 0.$$

□