Unsupervised Feature Selection via Graph Regularized Nonnegative CP Decomposition

Bilian Chen, Jiewen Guan, and Zhening Li

Abstract—Unsupervised feature selection has attracted remarkable attention recently. With the development of data acquisition technology, multi-dimensional tensor data has been appeared in enormous real-world applications. However, most existing unsupervised feature selection methods are non-tensor-based which results the vectorization of tensor data as a preprocessing step. This seemingly ordinary operation has led to an unnecessary loss of the multi-dimensional structural information and eventually restricted the quality of the selected features. To overcome the limitation, in this paper, we propose a novel unsupervised feature selection model: Nonnegative tensor CP (CANDECOMP/PARAFAC) decomposition based unsupervised feature selection, CPUFS for short. In specific, we devise new tensor-oriented linear classifier and feature selection matrix for CPUFS. In addition, CPUFS simultaneously conducts graph regularized nonnegative CP decomposition and newly-designed tensor-oriented pseudo label regression and feature selection to fully preserve the multi-dimensional data structure. To solve the CPUFS model, we propose an efficient iterative optimization algorithm with theoretically guaranteed convergence, whose computational complexity scales linearly in the number of features. A variation of the CPUFS model by incorporating nonnegativity into the linear classifier, namely CPUFSnn, is also proposed and studied. Experimental results on ten real-world benchmark datasets demonstrate the effectiveness of both CPUFS and CPUFSnn over the state-of-the-arts.

Index Terms—Unsupervised feature selection, nonnegative CP decomposition, optimization algorithm, classification

1 Introduction

Statistical learning algorithms often deal with data in high dimensional spaces. Due to the *curse of dimensionality* [1], these algorithms need a plethora of data to maintain their performance. In addition, a lot of redundancy and noise that inherently reside in the original high dimensional feature space will lead to confusion and degradation of learning systems [2]. To this end, the feature selection technique, which aims to select a small number of highly informative features, has been extensively studied in the past few decades. According to the availability of data labels, feature selection algorithms can be classified into three categories, namely supervised feature selection [3], [4], semi-supervised feature selection [5], [6] and unsupervised feature selection [7], [8]. Since data labels are usually unavailable or expensive in reality, unsupervised feature selection has a wider range of applications, but as a price, it is more challenging due to the lack of guidance from real data labels [9]. In this paper, we focus on the challenging unsupervised feature selection.

Unsupervised feature selection methods can be further grouped into three families, i.e., wrapper, embedded, and filter methods [10]. Among these methods, filter methods have better versatility since they select informative features according to the inner nature of data directly, regardless of the adopted statistical learning algorithm. On the other hand, wrapper methods utilize another independent predictive model to select the best performing features for that specific model, and the embedded methods perform feature selection and model learning at the same time. Our focus in this paper is on filter methods for unsupervised feature selection.

Most real-world data usually have multi-dimensional representation structures, i.e., they are *tensors* in contrast to vectors. For instance, images are two-dimensional, videos are three-dimensional, and fMRI data are usually four-dimensional [11]. In order to process multi-dimensional data, almost all existing unsupervised feature selection methods first vectorize data samples into one-dimensional vectors, and then conduct feature selection based on the vectorized data. However, the intrinsic multi-dimensional data structure will be discarded by the vectorization. This will lead to an unnecessary loss of information and eventually impair the performance of unsupervised feature selection. Nevertheless, there are almost no tensor-based unsupervised feature selection methods in the literature, to the best of our knowledge.

To avoid the aforementioned negative affect, we incorporate the tensor CP decomposition approach [12] and propose a novel unsupervised feature selection method, called nonnegative tensor CP decomposition based unsupervised

- *Bilian Chen and Jiewen Guan are with the Department of Automation, Xiamen University, Xiamen 361005, China, and also with the Xiamen Key Laboratory of Big Data Intelligent Analysis and Decision-Making, Xiamen 361005, China. E-mail: blchen@xmu.edu.cn, jwguan@stu.xmu.edu.cn.*
- *Zhening Li is with the School of Mathematics and Physics, University of Portsmouth, Portsmouth PO1 3HF, U.K. E-mail: zheningli@gmail.com.*

Manuscript received 9 May 2021; revised 23 January 2022; accepted 13 March 2022. Date of publication 17 March 2022; date of current version 6 January 2023.
This work was supported in part by the Youth Innovation Fund of Xiamen under Grant 3502Z20206049 and in part by the National Natural Science Foundation of China under Grant 61836005.
(Corresponding author: Bilian Chen.)
Recommended for acceptance by H. Ling.
Digital Object Identifier no. 10.1109/TPAMI.2022.3160205

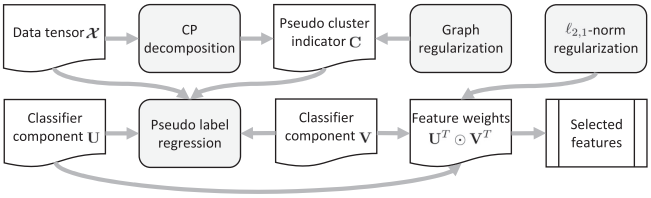

Data tensor $\mathcal{X}$
CP decomposition
Pseudo cluster indicator $\mathbf{C}$
Graph regularization
$\ell_{2,1}$-norm regularization
Classifier component $\mathbf{U}$
Pseudo label regression
Classifier component $\mathbf{V}$
Feature weights $\mathbf{U}^T \odot \mathbf{V}^T$
Selected features

Fig. 1. The framework of CPUFS.

feature selection (CPUFS), for selecting features from two-dimensional data. Specifically, we 1) adopt graph regularized nonnegative CP decomposition [13], [14] to generate pseudo class labels, and 2) design new linear classifier and feature selection matrix tailored for tensor data to perform pseudo label regression and feature selection. In this way, the multi-dimensional structural information of data can be well preserved in every part of the whole feature selection process. In terms of solving the proposed CPUFS model, we put forward an efficient iterative optimization algorithm with guaranteed convergence. In addition, the computational complexity of our proposed algorithm scales linearly in the number of features, which guarantees the efficiency of feature selection. To further consider the nonnegativity of the input tensor data and the generated pseudo class labels, we also propose a variation of CPUFS, named CPUFSnn, which additionally imposes a nonnegativity constraint on the linear classifier. Due to the nonnegative linear classifier, CPUFSnn may select higher quality features. It is also worth mentioning that, our proposed methods that apply to two-dimensional data can be easily extended to three or higher dimensional cases. The framework of our CPUFS method is illustrated in Fig. 1.

Discussions: This is the first treatise to utilize tensor decomposition technique for unsupervised feature selection, to the best of our knowledge. Although it seems that the extension for unsupervised feature selection from the matrix-based to the tensor-based is standard, the underlying challenges are far from trivial, due to the following reasons.

1) *Preserving multi-dimensional structural information of tensor data in every step of the whole feature selection process can be easily overlooked.* An accidental negligence can directly lead to an unnecessary loss of information. In order to avoid the loss, we incorporate nonnegative tensor CP decomposition and propose new linear classifier and feature selection matrix.
2) *The design of the linear classifier requires a careful treatment.* Distinct from the existing methods which vectorize the data to perform pseudo label regression, we design a new linear classifier that involves multiple parameter matrices (see Section 4.1) to preserve the multi-dimensional data structure.
3) *The design of the feature selection matrix is challenging*. As aforementioned, the newly-designed linear classifier involves multiple parameter matrices. Therefore, it is highly nontrivial and indeed important to design a feature selection matrix to integrate these parameter matrices together, reserve the preserved information inside them, and serve as the importance weights of features. The new feature selection matrix that we design (see Section 4.1) is able to achieve the aforementioned three points. Moreover, this can also be extended to higher dimensional cases.

We highlight our main contributions as below.

1) We propose a novel nonnegative tensor CP decomposition based unsupervised feature selection model, CPUFS, that can preserve the multi-dimensional data structure in the whole feature selection process. A finer model, CPUFSnn, is also proposed by considering the nonnegativity of the input tensor data and the generated pseudo class labels.
2) We develop an efficient iterative optimization algorithm with theoretically guaranteed convergence to solve CPUFS and CPUFSnn models. Moreover, its computational complexity scales linearly in the number of features.
3) We evaluate the proposed CPUFS and CPUFSnn models on ten real-world benchmark datasets. Experimental results show that our methods outperform the state-of-the-arts. In addition, we investigate parameter sensitivity, running time, empirical convergence speed and distribution of the selected features of CPUFS.

The rest of this paper is organized as follows. First, we review relevant work in Section 2 and introduce related preliminaries in Section 3. Then, we present our proposed CPUFS model and its optimization algorithm as well as its variation in Section 4. We report the experimental results in Section 5 and conclude this paper in Section 6.

2 Related Work

In this section, we first review relevant non-tensor-based unsupervised feature selection methods, followed by tensor-based feature extraction and selection methods, with an emphasis on the former.

2.1 Non-Tensor-Based Unsupervised Feature Selection

In recent decades, unsupervised feature selection methods have been extensively studied. Almost all of them are non-tensor-based and we comprehensively review relevant ones in the chronological order.

- *Laplacian score* (LS) [15] evaluates features separately according to their capabilities to preserve the local geometrical structure of data.
- *Multi-cluster feature selection* (MCFS) [16] adopts joint spectral analysis and ℓ_1-norm regularized regression to select informative features that best preserve the multi-cluster structure of data.
- *Unsupervised discriminative feature selection* (UDFS) [17] picks the most discriminative features by simultaneously conducting $\ell_{2,1}$-norm based feature selection and local discriminative analysis.

- *Feature selection via joint embedding learning and sparse regression* (JELSR) [18] adopts locally linear approximation to construct similarity graph and unifies embedding learning and sparse regression to perform feature selection.
- *Nonnegative discriminative feature selection* (NDFS) [19] jointly exploits correlation between features and local discriminative information of data.
- *Robust unsupervised feature selection* (RUFS) [20] simultaneously performs robust nonnegative matrix factorization based pseudo class label learning and feature selection.
- *Robust spectral analysis for unsupervised feature selection* (RSFS) [21] performs spectral analysis and sparse spectral regression at the same time under a robust joint framework.
- *Simultaneous orthogonal basis clustering and feature selection* (SOCFS) [22] uses a new type of target matrix to perform orthogonal basis clustering that guides the process of feature selection.
- *Unsupervised feature selection with adaptive structure learning* (FSASL) [23] and *structured optimal graph feature selection* (SOGFS) [24] jointly perform $\ell_{2,1}$-norm based feature selection and local geometrical structure optimizing.
- *Coupled dictionary learning for unsupervised feature selection* (CDLFS) [25] reconstructs data by dictionary learning and uses the learned representation coefficients to model data distribution, which will be further used for feature selection.
- *Generalized uncorrelated regression with adaptive graph for unsupervised feature selection* (URAFS) [26] simultaneously performs uncorrelated regression model learning, $\ell_{2,1}$-norm based feature selection and maximum entropy based adaptive graph structure learning.
- *Dependence guided unsupervised feature selection* (DGUFS) [27] jointly conducts feature selection and data clustering while increasing the inter-dependence among the original data, the generated pseudo labels and the selected features.
- *Unsupervised feature selection with row-sparsity constraint and optimized graph* (RSOGFS) [7] performs $\ell_{2,0}$-norm based feature selection and adaptive similarity graph learning at the same time.
- *Sparse principal component analysis for feature selection* (SPCAFS) [8] imposes an $\ell_{2,p}$-norm based sparsity regularization on the projection matrix of principal component analysis (PCA) [28] for feature selection.

Discussions: All the above-mentioned methods are non-tensor-based and so they are relative simple to implement. However, as a price, the multi-dimensional structural information of tensor data was lost among all these methods since they vectorize the data to make them processable. In contrast, our proposed CPUFS and CPUFSnn models adopt the nonnegative tensor CP decomposition for generating pseudo class labels and utilize newly-designed linear classifier and feature selection matrix for pseudo label regression and feature selection. They fully preserve the multi-dimensional structural information in tensor data.

2.2 Tensor-Based Feature Extraction and Selection

2.2.1 Tensor-Based Feature Extraction

Tensor-based feature extraction methods have also attracted considerable attention recently. However, few of these are performed on unsupervised feature selection. We first briefly review tensor-based feature extraction methods. Although these methods are not oriented for unsupervised feature selection, they may bring some insightful inspirations to the development of tensor-based feature selection methods.

- *Multilinear principal component analysis* (MPCA) [29] generalizes classical PCA [28] to higher-order cases to extract features from tensor data.
- *Manifold regularization nonnegative Tucker decomposition* (MR-NTD) [30] simultaneously conduct nonnegative Tucker decomposition and Laplacian regularization to extract core feature tensors which preserve the local geometrical structure.
- *Tensor robust principal component analysis* (TRPCA) [31] extends robust PCA [32] to the tensor case to extract features from tensor data while considering outliers and noise of data. It is similar to MPCA.
- *Tensor Bayesian vectorial dimension reduction* (TBV-DR) [33] represents a tensor as a linear combination of some tensor bases in the same order with the coefficient vector being taken as the dimension-reduced representation.
- *Low-rank tensor decomposition with feature variance maximization* (TDVM) [34] extracts features from incomplete tensor data via simultaneously estimating missing entry and exploring feature relationships.

2.2.2 Tensor-Based Unsupervised Feature Selection

We review the only tensor-based unsupervised feature selection in the literature, to the best of our knowledge.

- *Graph regularized low-rank tensor representation* (GRLTR) [35] is an embedded feature selection method that performs low-rank tensor representation learning, local geometrical structure learning and $\ell_{2,1}$-norm based feature selection simultaneously.

Discussions: Although GRLTR is a tensor-based method, the multi-dimensional structural information of data is lost in the manifold projection process. This is because GRLTR uses the mode-n matricization to unfold tensor data and then projects the unfolded data into a low-dimensional manifold. The manifold projection process is actually the most important part of feature selection since the projection matrix is also the feature selection matrix. In contrast, our proposed CPUFS and CPUFSnn models take into account the multi-dimensional data structural information in not only the pseudo label generation process but also the pseudo label regression and feature selection processes.

3 Preliminaries

Before presenting our main mathematical models, we introduce some notations and terminologies used in the paper.

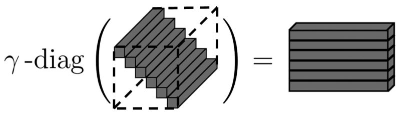


Fig. 2. Illustration of γ-diag.

3.1 Notations

Throughout this paper, we use boldface calligraphic letters to denote tensors, boldface capital letters to denote matrices, boldface lowercase letters to denote vectors, and italic lowercase letters to denote scalars. An element of a vector $\mathbf{x}$, a matrix $\mathbf{X}$, and a tensor $\boldsymbol{\mathcal{X}}$ is denoted by x_i, x_{ij}, and x_{ijk}, respectively, depending on the number of indices (a.k.a., modes).

For a matrix $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2}$, $\mathbf{x}_{i:}$ and $\mathbf{x}_{:j}$ represent the ith row and the jth column vector of $\mathbf{X}$, respectively. We denote $\mathrm{Tr}(\mathbf{X})$ to be the trace of $\mathbf{X}$ if it is squared, $\mathbf{X}^T$ to be the transpose of $\mathbf{X}$, and $\mathbf{X}_+$ to be the nonnegative part of $\mathbf{X}$, i.e., zeros out negative entries. The Frobenius norm of $\mathbf{X}$ is denoted by $\|\mathbf{X}\|_F$ and the $\ell_{2,1}$-norm is denoted by $\|\mathbf{X}\|_{2,1}$. The Kronecker product is denoted by $\otimes$, the Khatri-Rao product is denoted by $\odot$, and the Hadamard product is denoted by $\circledast$. Besides, we use $\mathbf{I}_n$ to denote the $n \times n$ identity matrix, $\mathbf{I}_{m,n}$ to denote the top-left $m \times n$ truncation of an identity matrix, and $\mathbf{1}_n$ to denote the n-dimensional all-one vector.

For a third-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, the vector along the first mode is called the mode-1 fiber (denoted by $\mathbf{x}_{:ij}$ for the (i,j)th one), the vector along the second mode is called the mode-2 fiber (denoted by $\mathbf{x}_{i:j}$), and the vector along the third mode is called the mode-3 fiber (denoted by $\mathbf{x}_{ij:}$). The matrix $\mathbf{X}_{(i)}$ is called the matricization of $\boldsymbol{\mathcal{X}}$ along the ith mode and can be constructed by arranging the mode-i fibers to be the columns of the resulting matrix. The mode-i (matrix) product of $\boldsymbol{\mathcal{X}}$ with a matrix $\mathbf{U}$ is denoted by $\boldsymbol{\mathcal{X}} \times_i \mathbf{U}$. Same as a matrix, the Frobenius norm of $\boldsymbol{\mathcal{X}}$ is denoted by $\|\boldsymbol{\mathcal{X}}\|_F$. For convenience, we use $\mathbf{X}_{::i}$ and $\mathbf{X}^{(i)}$ interchangeably to denote the ith frontal slice of $\boldsymbol{\mathcal{X}}$. The readers may refer to [12] for more details on tensor notations and operations. We summarize frequently used notations in Table 1.

Definition 3.1 (3D tensor diagonal elements). *For a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{j \times j \times k}$ whose frontal slices are square matrices, we use γ-diag$(\boldsymbol{\mathcal{X}}) \in \mathbb{R}^{j \times k}$ to denote the diagonal elements of $\boldsymbol{\mathcal{X}}$, where the rows of γ-diag$(\boldsymbol{\mathcal{X}})$ are composed of the mode-3 fibers of $\boldsymbol{\mathcal{X}}$ in the form $\mathbf{x}_{ii:}$ for $i = 1, 2, \ldots, j$, i.e.,*

$$\gamma\text{-diag}(\boldsymbol{\mathcal{X}}) = \begin{bmatrix} \mathbf{x}_{11:}^T \\ \mathbf{x}_{22:}^T \\ \vdots \\ \mathbf{x}_{jj:}^T \end{bmatrix} = \begin{bmatrix} x_{111} & x_{112} & \cdots & x_{11k} \\ x_{221} & x_{222} & \cdots & x_{22k} \\ \vdots & \vdots & \ddots & \vdots \\ x_{jj1} & x_{jj2} & \cdots & x_{jjk} \end{bmatrix}.$$

The γ-diag operator is illustrated in Fig. 2.

Definition 3.2 (Inverse vectorization). *We use $\mathrm{vec}_{m,n}^{-1}$ to denote the inverse vectorization mapping from vectors in $\mathbb{R}^{mn \times 1}$ to matrices in $\mathbb{R}^{m \times n}$, i.e.,*

$$\mathbf{x} \mapsto \mathrm{vec}_{m,n}^{-1}(\mathbf{x}) = \Big((\mathrm{vec}(\mathbf{I}_n))^T \otimes \mathbf{I}_m \Big) (\mathbf{I}_n \otimes \mathbf{x}).$$

TABLE 1
Summary of Frequently-Used Notations

x	A scalar	$\mathbf{x}$	A vector
$\mathbf{X}$	A matrix	$\boldsymbol{\mathcal{X}}$	A tensor
$\mathbf{X}^T$	The transpose of $\mathbf{X}$	$\mathrm{Tr}\,(\mathbf{X})$	$\mathrm{Tr}\,(\mathbf{X}) = \sum_i x_{ii}$
$\|\mathbf{X}\|_F$	$\|\mathbf{X}\|_F = \sqrt{\sum_{ij} x_{ij}^2}$	$\|\mathbf{X}\|_{2,1}$	$\|\mathbf{X}\|_{2,1} = \sum_i \sqrt{\sum_j x_{ij}^2}$
$\mathbf{X}_{::i}$	The ith frontal slice of $\boldsymbol{\mathcal{X}}$	$\mathbf{X}^{(i)}$	$\mathbf{X}^{(i)} = \mathbf{X}_{::i}$
$\mathbf{x}_{ij:}$	The (i,j)th mode-3 fiber of $\boldsymbol{\mathcal{X}}$	$\mathbf{X}_{(i)}$	The mode-i matricization of $\boldsymbol{\mathcal{X}}$
$\boldsymbol{\mathcal{X}} \times_i \mathbf{U}$	The mode-i product of $\boldsymbol{\mathcal{X}}$ and $\mathbf{U}$	$\|\boldsymbol{\mathcal{X}}\|_F$	$\|\boldsymbol{\mathcal{X}}\|_F = \sqrt{\sum_{ijk} x_{ijk}^2}$
$\otimes$	The Kronecker prduct	$\odot$	The Khatri-Rao product
$\circledast$	The Hadamard product		

Intuitively, the elements in $\mathrm{vec}_{m,n}^{-1}(\mathbf{x})$ maintains their column-wise order from $\mathbf{x}$. As an example, for a vector $\mathbf{x}_0 = [1,2,3,4,5,6,7,8]^T \in \mathbb{R}^{8 \times 1}$, its inverse vectorization $\mathrm{vec}_{2,4}^{-1}(\mathbf{x}_0) = \begin{bmatrix} 1 & 3 & 5 & 7 \\ 2 & 4 & 6 & 8 \end{bmatrix} \in \mathbb{R}^{2 \times 4}$.

3.2 Data Assumptions

Assume that the dataset $\mathcal{D} = \{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n\}$ consists of n samples and each sample $\mathbf{X}_i \in \mathbb{R}^{n_1 \times n_2}$ is represented by a matrix. The input of our CPUFS method is a data tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times n_2 \times n}$ whose frontal slices $\mathbf{X}_{::i} = \mathbf{X}_i$ for $i = 1, 2, \ldots, n$. CPUFS aims at ranking these $n_1 n_2$ features so that the selection of any specific number of features can be conducted. Assume that the dataset $\mathcal{D}$ is naturally divided into c classes with each containing at least one sample. We then introduce a Boolean matrix $\mathbf{Y} \in \{0,1\}^{n \times c}$ to describe the sample-cluster memberships of $\mathcal{D}$. That is, if $\mathbf{X}_i$ belongs to the jth cluster, then $y_{ij} = 1$; otherwise, $y_{ij} = 0$. Following [17], [19], we define the scaled cluster indicator matrix as $\mathbf{C} = \mathbf{Y}(\mathbf{Y}^T\mathbf{Y})^{-1/2}$, which has better properties and serves as a variable to be optimized in our CPUFS model. Notice that $\mathbf{C}^T\mathbf{C} = (\mathbf{Y}^T\mathbf{Y})^{-1/2}\mathbf{Y}^T\mathbf{Y}(\mathbf{Y}^T\mathbf{Y})^{-1/2} = \mathbf{I}_c$.

3.3 Graph Regularization

One of the key points in unsupervised feature selection is how to preserve the local geometrical structure of data [19], [23]. This is specifically expressed as forcing geometrically close samples to have consistent embeddings, which, in our case, are scaled cluster indicators $\mathbf{c}_{:i}$'s [13]. According to [13], the local geometrical structure can be effectively modeled through a weighted k-nearest neighbor graph, where the nodes represent data samples and the edges represent the affinity between data samples. In particular, following [13], we can construct a weighted k-nearest neighbor graph whose adjacency matrix $\mathbf{S}$ is computed by

$$s_{ij} = \begin{cases} \exp\left(\frac{\|\mathbf{X}_i - \mathbf{X}_j\|_F^2}{-\sigma^2} \right) & \text{if } \mathbf{X}_i \in \mathcal{N}_k(\mathbf{X}_j) \text{ or } \mathbf{X}_j \in \mathcal{N}_k(\mathbf{X}_i) \\ 0 & \text{otherwise,} \end{cases}$$

where $\mathcal{N}_k(\mathbf{X}_i)$ denotes the set of k-nearest neighbors of $\mathbf{X}_i$ and σ is the Gaussian kernel width. With the above constructed weighted k-nearest neighbor graph, the local geometrical structure can be effectively preserved by minimizing the following objective function [19], [36]

$$\min_{\mathbf{C}} \ \frac{1}{2} \sum_{i,j=1}^{n} s_{ij} \left\| \frac{\mathbf{c}_{:i}}{\sqrt{d_{ii}}} - \frac{\mathbf{c}_{:j}}{\sqrt{d_{jj}}} \right\|_2^2 = \mathrm{Tr}\big(\mathbf{C}^T\mathbf{L}\mathbf{C}\big),$$

where $\mathbf{L} = \mathbf{I}_n - \mathbf{D}^{-1/2}\mathbf{S}\mathbf{D}^{-1/2}$ is the normalized graph Laplacian of $\mathbf{S}$ and $\mathbf{D} = \mathrm{diag}(\mathbf{S}\mathbf{1}_n)$ is the degree matrix of $\mathbf{S}$.

3.4 CP and Nonnegative CP Decompositions

CP decomposition [14] is the most widely used decomposition of tensors. For a third-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, CP decomposition is to decompose $\boldsymbol{\mathcal{X}}$ as

$$\boldsymbol{\mathcal{X}} \approx \sum_{r=1}^{r_0} \mathbf{a}_{:r} \circ \mathbf{b}_{:r} \circ \mathbf{c}_{:r} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!],$$

where $\circ$ represents the vector outer product, $\mathbf{A} \in \mathbb{R}^{n_1 \times r_0}$, $\mathbf{B} \in \mathbb{R}^{n_2 \times r_0}$, $\mathbf{C} \in \mathbb{R}^{n_3 \times r_0}$, r_0 is the desired rank, and the last equality uses the shorthand $[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ introduced in [37].

The nonnegative CP decomposition is a special case of CP decomposition in which the given tensor $\boldsymbol{\mathcal{X}}$ is (entry-wise) nonnegative and all the factor matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are required to be nonnegative. Besides, it also has equivalence to the k-means clustering [38]. Usually, solving the nonnegative CP decomposition resorts to minimizing the Frobenius norm approximation $\|\boldsymbol{\mathcal{X}} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]\|_F^2$ while keeping $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ nonnegative.

4 Nonnegative CP Decomposition Based Unsupervised Feature Selection

4.1 Tensor-Oriented Linear Classifier and Feature Selection Matrix Design

We first propose a new tensor-oriented linear classifier and feature selection matrix for CPUFS.

Linear classifier design. Since the tensor data is multi-dimensional, we use the tensor mode-n product to design tensor-oriented linear classifier[1] in order to preserve the multi-dimensional data structure. Specifically, given a data tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times n_2 \times n}$, we naturally design the linear classifier as γ-diag$(\boldsymbol{\mathcal{X}} \times_1 \mathbf{U} \times_2 \mathbf{V})$ where $\mathbf{U} \in \mathbb{R}^{c \times n_1}$ and $\mathbf{V} \in \mathbb{R}^{c \times n_2}$ are two collaborative components. That is, for every $j = 1, 2, \ldots, c$, $\mathbf{u}_{j:}$ and $\mathbf{v}_{j:}$ are collaboratively and inseparably responsible for fitting the jth scaled cluster indicator $\mathbf{c}_{:j}$ by $[\mathbf{u}_{j:}\mathbf{X}^{(1)}\mathbf{v}_{j:}^T, \mathbf{u}_{j:}\mathbf{X}^{(2)}\mathbf{v}_{j:}^T, \ldots, \mathbf{u}_{j:}\mathbf{X}^{(n)}\mathbf{v}_{j:}^T]^T$. In such a way, the multi-dimensional data structure can be well preserved.

Feature selection matrix design. The feature selection matrix shall be designed based on the parameter matrices $\mathbf{U}$ and $\mathbf{V}$ of the linear classifier, while reversing their multi-dimensional information and serving as the importance weights of features. Specifically, we design an innovative feature selection matrix $(\mathbf{U}^T \odot \mathbf{V}^T) \in \mathbb{R}^{n_1 n_2 \times c}$. The reason is as follows. Let us denote $(\gamma\text{-diag}(\boldsymbol{\mathcal{X}} \times_1 \mathbf{U} \times_2 \mathbf{V}))^T$ to be $\hat{\mathbf{C}}$ and consider its element $\hat{c}_{ij}$, the prediction from the linear classifier indicating whether the ith sample $\mathbf{X}^{(i)}$ belongs to the jth cluster or not. It is calculated that

$$\hat{c}_{ij} = \mathbf{u}_{j:}\mathbf{X}^{(i)}\mathbf{v}_{j:}^T = \sum_{h=1}^{n_1}\sum_{g=1}^{n_2} u_{jh} v_{jg} x_{hgi}.$$

In this expression, x_{hgi} represents the value of the (h, g)th feature of $\mathbf{X}^{(i)}$ and $u_{jh}v_{jg}$ can be interpreted as the importance weight of the (h, g)th feature for the jth cluster as it is irrelevant to the sample index i. Coincidentally, we notice that the vector $[u_{1h}v_{1g}, u_{2h}v_{2g}, \ldots, u_{ch}v_{cg}]$ happens to be the $((h-1)n_2 + g)$th row of $(\mathbf{U}^T \odot \mathbf{V}^T)$! Besides, the mapping $(h, g) \mapsto (h-1)n_2 + g$ from $\mathbb{Z}_+ \times \mathbb{Z}_+$ to $\mathbb{Z}_+$ is obviously a *bijection*. Therefore, we design the feature selection matrix as $\mathbf{U}^T \odot \mathbf{V}^T$.

Discussions: It is worth mentioning that the importance weight of the (h, g)th feature for the jth cluster is composed of two parts, u_{jh} and v_{jg}. Each part controls the importance weight of the hth row and the gth column respectively and independently. Therefore, for features from the same row or column, the corresponding importance weights are directly correlated. This mechanism endows the importance weights of features with the multi-dimensional structural characteristics. We will show how this mechanism appears in reality in Section 5.6.

4.2 The Optimization Model

The main tools for developing CPUFS have been already introduced or established in Section 3 and Section 4.1. We now propose the CPUFS model, followed by detailed explanations for each term and each constraint. The optimization model is formulated as

$$\begin{aligned}\min_{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{U},\mathbf{V},\mathbf{Y}} \ & \|\boldsymbol{\mathcal{X}} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]\|_F^2 + \nu \mathrm{Tr}\big(\mathbf{C}^T\mathbf{L}\mathbf{C}\big) \\ & + \alpha\Big\|\big(\gamma\text{-diag}(\boldsymbol{\mathcal{X}} \times_1 \mathbf{U} \times_2 \mathbf{V})\big)^T - \mathbf{C}\Big\|_F^2 + \beta\big\|\mathbf{U}^T \odot \mathbf{V}^T\big\|_{2,1} \\ \text{s.t.} \ & \mathbf{A}, \mathbf{B}, \mathbf{C} \geq 0, \mathbf{C} = \mathbf{Y}\big(\mathbf{Y}^T\mathbf{Y}\big)^{-\frac{1}{2}}, \mathbf{Y} \in \{0, 1\}^{n \times c}, \end{aligned} \qquad (1)$$

where ν, α and β are parameters to balance different terms in the objective function, c and n are the number of latent classes and data samples, respectively, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times n_2 \times n}$, $\mathbf{A} \in \mathbb{R}^{n_1 \times c}$, $\mathbf{B} \in \mathbb{R}^{n_2 \times c}$, $\mathbf{C} \in \mathbb{R}^{n \times c}$, $\mathbf{L} \in \mathbb{R}^{n \times n}$, $\mathbf{U} \in \mathbb{R}^{c \times n_1}$ and $\mathbf{V} \in \mathbb{R}^{c \times n_2}$ are defined similarly as in Section 3.2, Section 3.3, Section 3.4 and Section 4.1. The explanations of each term and constraint in (1) are as follows.

- $\|\boldsymbol{\mathcal{X}} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]\|_F^2$ with $\mathbf{A}, \mathbf{B}, \mathbf{C} \geq 0$ is the standard nonnegative tensor CP decomposition of data tensor $\boldsymbol{\mathcal{X}}$. It is responsible for generating pseudo class labels $\mathbf{C}$ while preserving the multi-dimensional structural information.
- $\mathrm{Tr}(\mathbf{C}^T\mathbf{L}\mathbf{C})$ is the graph regularizer. It aims to capture the intrinsic geometrical structure of data so that the generated pseudo class labels of similar data samples are kept as consistent as possible.
- $\|(\gamma\text{-diag}(\boldsymbol{\mathcal{X}} \times_1 \mathbf{U} \times_2 \mathbf{V}))^T - \mathbf{C}\|_F^2$ is the pseudo label regression of our newly-designed linear classifier γ-diag$(\boldsymbol{\mathcal{X}} \times_1 \mathbf{U} \times_2 \mathbf{V})$ to the generated pseudo class labels $\mathbf{C}$. The Frobenius norm is adopted to measure the discrepancy.
- $\|\mathbf{U}^T \odot \mathbf{V}^T\|_{2,1}$ is the $\ell_{2,1}$-norm based row-sparsity regularizer for the feature selection matrix $\mathbf{U}^T \odot \mathbf{V}^T$. The $\ell_{2,1}$-norm imposed on the matrix forces its rows to be sparse. As the rows of the feature selection matrix encode the importance weights of different features, the smaller the ℓ_2-norm of a certain row, the less the correlation between the corresponding feature and the pseudo cluster labels $\mathbf{C}$, and hence the less importance of that feature.

[1] Note that the tensor contracted product [39] cannot be adopted here as otherwise the multi-dimensional data structure would be lost.

- $\mathbf{C} = \mathbf{Y}(\mathbf{Y}^T\mathbf{Y})^{-\frac{1}{2}}$ with $\mathbf{Y} \in \{0,1\}^{n\times c}$ is a combinatorial constraint that enforces $\mathbf{C}$ to meet the requirements of being a scaled cluster indicator matrix.

Generally speaking, the first two terms in the objective function of (1) try to learn pseudo cluster labels while preserving the local geometrical structure of data whereas the last two terms aim at fitting learned pseudo cluster labels by a linear classifier with sparsity regularization. The two parts can guide each other in the iterative optimization process. However, due to the discrete nature of the constraints, (1) is very hard to be solved. In order to solve the problem, some relaxation and reformulation are needed and we will discuss these in the next subsection.

4.3 Solution Method

4.3.1 CPUFS Reformulation

The most difficult part of (1) lies in the binary constraint $\mathbf{C} = \mathbf{Y}(\mathbf{Y}^T\mathbf{Y})^{-\frac{1}{2}}$ with $\mathbf{Y} \in \{0,1\}^{n\times c}$. One common approach to overcome this is to relax $\mathbf{C}$ as a continuous variable. However, directly relaxing $\mathbf{C}$ to be continuous may bring severe negative effects. For instance, the learned $\mathbf{C}$ might be a very dense matrix and deteriorate its cluster indicating precision. Notice that $\mathbf{C}$ has an important property, i.e., $\mathbf{C}^T\mathbf{C} = \mathbf{I}_c$ (see Section 3.2), which can get rid of most undesirable solutions. Therefore, we first relax (1) to be

$$\begin{aligned}\min_{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{U},\mathbf{V}} & \|\boldsymbol{\mathcal{X}} - [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]\|_F^2 + \nu\mathrm{Tr}\big(\mathbf{C}^T\mathbf{L}\mathbf{C}\big)\\ & + \alpha\Big\|\big(\gamma\text{-diag}(\boldsymbol{\mathcal{X}}\times_1\mathbf{U}\times_2\mathbf{V})\big)^T - \mathbf{C}\Big\|_F^2 + \beta\big\|\mathbf{U}^T\odot\mathbf{V}^T\big\|_{2,1}\\ \text{s.t.}\quad & \mathbf{A},\mathbf{B},\mathbf{C}\geq 0, \mathbf{C}^T\mathbf{C} = \mathbf{I}_c.\end{aligned}$$

Notice that the nonnegativity and orthogonality simultaneously imposed on $\mathbf{C}$ still make the objective function hard to optimize. Similar to [22], we propose another equivalent optimization problem

$$\begin{aligned}\min_{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{F},\mathbf{U},\mathbf{V}} & \|\boldsymbol{\mathcal{X}} - [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]\|_F^2 + \nu\mathrm{Tr}\big(\mathbf{C}^T\mathbf{L}\mathbf{F}\big)\\ & + \alpha\Big\|\big(\gamma\text{-diag}(\boldsymbol{\mathcal{X}}\times_1\mathbf{U}\times_2\mathbf{V})\big)^T - \mathbf{F}\Big\|_F^2 + \beta\big\|\mathbf{U}^T\odot\mathbf{V}^T\big\|_{2,1}\\ \text{s.t.}\quad & \mathbf{A},\mathbf{B},\mathbf{F}\geq 0, \mathbf{F} = \mathbf{C}, \mathbf{C}^T\mathbf{C} = \mathbf{I}_c,\end{aligned}$$

where $\mathbf{F}$ is an auxiliary matrix to alleviate the difficulty of optimization. By adding a penalty function $\eta\|\mathbf{C} - \mathbf{F}\|_F^2$ which describes the cost of violating the constraint $\mathbf{F} = \mathbf{C}$ to the objective function, we can reformulate the above optimization problem to be

$$\begin{aligned}\min_{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{F},\mathbf{U},\mathbf{V}} & \|\boldsymbol{\mathcal{X}} - [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]\|_F^2 + \nu\mathrm{Tr}\big(\mathbf{C}^T\mathbf{L}\mathbf{F}\big) + \eta\|\mathbf{C} - \mathbf{F}\|_F^2\\ & + \alpha\Big\|\big(\gamma\text{-diag}(\boldsymbol{\mathcal{X}}\times_1\mathbf{U}\times_2\mathbf{V})\big)^T - \mathbf{F}\Big\|_F^2 + \beta\big\|\mathbf{U}^T\odot\mathbf{V}^T\big\|_{2,1}\\ \text{s.t.}\quad & \mathbf{A},\mathbf{B},\mathbf{F}\geq 0, \mathbf{C}^T\mathbf{C} = \mathbf{I}_c,\end{aligned}\tag{2}$$

where η is a large constant to ensure that $\mathbf{F}$ and $\mathbf{C}$ are sufficiently close. This reformulation has greatly reduced the difficulty to solve the optimization model as each matrix variable has at most one constraint. In what follows, we propose an efficient algorithm for the reformulated model.

4.3.2 Optimization Algorithm Design

The optimization model (2) remains nonconvex and cannot be solved directly albeit its difficulty has been greatly alleviated. However, due to the separable nature of the six matrix variables, it can be solved one by one cyclically by fixing others and iteratively improved. Therefore, we propose the following iterative algorithm via alternative updating.

Updating $\mathbf{A}$*:* Updating $\mathbf{A}$ with other variables fixed leads to a standard NMF formulation [40]

$$\min_{\mathbf{A}\geq 0} \ \Big\|\mathbf{X}_{(1)} - \mathbf{A}(\mathbf{C}\odot\mathbf{B})^T\Big\|_F^2,$$

whose updating rule is

$$\mathbf{A} \leftarrow \mathbf{A}\circledast\frac{\mathbf{X}_{(1)}(\mathbf{C}\odot\mathbf{B})}{\mathbf{A}(\mathbf{C}\odot\mathbf{B})^T(\mathbf{C}\odot\mathbf{B})},$$

where the division is performed entry-wisely.

Updating $\mathbf{B}$*:* Similar to updating $\mathbf{A}$, updating $\mathbf{B}$ with other variables fixed also leads to a standard NMF formulation

$$\min_{\mathbf{B}\geq 0} \ \Big\|\mathbf{X}_{(2)} - \mathbf{B}(\mathbf{C}\odot\mathbf{A})^T\Big\|_F^2,$$

whose updating rule is

$$\mathbf{B} \leftarrow \mathbf{B}\circledast\frac{\mathbf{X}_{(2)}(\mathbf{C}\odot\mathbf{A})}{\mathbf{B}(\mathbf{C}\odot\mathbf{A})^T(\mathbf{C}\odot\mathbf{A})}.$$

Updating $\mathbf{C}$*:* The subproblem relevant to $\mathbf{C}$ is

$$\min_{\mathbf{C}^T\mathbf{C}=\mathbf{I}_c} \ \Big\|\mathbf{X}_{(3)} - \mathbf{C}(\mathbf{B}\odot\mathbf{A})^T\Big\|_F^2 + \nu\mathrm{Tr}\big(\mathbf{C}^T\mathbf{L}\mathbf{F}\big) + \eta\|\mathbf{C} - \mathbf{F}\|_F^2,$$

which can be equivalently reformulated as

$$\min_{\mathbf{C}^T\mathbf{C}=\mathbf{I}_c} \ \big\|\mathbf{C} - \big(2\mathbf{X}_{(3)}(\mathbf{B}\odot\mathbf{A}) - \nu\mathbf{L}\mathbf{F} + 2\eta\mathbf{F}\big)\big\|_F^2.$$

Lemma 4.1 (Orthogonal Procrustes problem [41]). *Given two matrices* $\mathbf{P}\in\mathbb{R}^{n\times m}$ *and* $\mathbf{Q}\in\mathbb{R}^{n\times d}$*, the optimization problem*

$$\min_{\tilde{\mathbf{T}}\tilde{\mathbf{T}}^T=\mathbf{I}_m} \big\|\mathbf{P}\tilde{\mathbf{T}} - \mathbf{Q}\big\|_F^2$$

has an analytical solution

$$\tilde{\mathbf{T}} = \mathbf{U}\mathbf{I}_{m,d}\mathbf{V}^T,$$

where $\mathbf{U}\in\mathbb{R}^{m\times m}$ *and* $\mathbf{V}\in\mathbb{R}^{d\times d}$ *are formed by the left and right singular vectors of* $\mathbf{P}^T\mathbf{Q}$ *via singular value decomposition, respectively.*

By letting $\mathbf{P} = \mathbf{I}_c$, $\tilde{\mathbf{T}} = \mathbf{C}^T$, and $\mathbf{Q} = 2(\mathbf{B}\odot\mathbf{A})^T\mathbf{X}_{(3)}^T - \nu\mathbf{F}^T\mathbf{L}^T + 2\eta\mathbf{F}^T$ in Lemma 4.1, the solution of the subproblem of updating $\mathbf{C}$ can be obtained as

$$\mathbf{C} \leftarrow \mathbf{V}_{\mathbf{C}}\mathbf{I}_{n,c}\mathbf{U}_{\mathbf{C}}^T,$$

where $\mathbf{U}_{\mathbf{C}}$ and $\mathbf{V}_{\mathbf{C}}$ are formed by the left and right singular vectors of $2(\mathbf{B}\odot\mathbf{A})^T\mathbf{X}_{(3)}^T - \nu\mathbf{F}^T\mathbf{L}^T + 2\eta\mathbf{F}^T$, respectively.

Updating $\mathbf{F}$*:* The subproblem relevant to $\mathbf{F}$ is

$$\min_{\mathbf{F}\geq 0} \ \alpha\Big\|(\gamma\text{-diag}(\boldsymbol{\mathcal{X}} \times_1 \mathbf{U} \times_2 \mathbf{V}))^T - \mathbf{F}\Big\|_F^2 + \nu\mathrm{Tr}\big(\mathbf{C}^T\mathbf{L}\mathbf{F}\big) + \eta\|\mathbf{C}-\mathbf{F}\|_F^2,$$

which can be equivalently reformulated as

$$\min_{\mathbf{F}\geq 0} \ \left\|\mathbf{F} - \frac{\alpha(\gamma\text{-diag}(\boldsymbol{\mathcal{X}} \times_1 \mathbf{U} \times_2 \mathbf{V}))^T + \eta\mathbf{C} - \frac{\nu}{2}\mathbf{L}^T\mathbf{C}}{\alpha+\eta}\right\|_F^2.$$

The optimal updating rule for $\mathbf{F}$ is obviously

$$\mathbf{F} \leftarrow \frac{\Big(\alpha(\gamma\text{-diag}(\boldsymbol{\mathcal{X}} \times_1 \mathbf{U} \times_2 \mathbf{V}))^T + \eta\mathbf{C} - \frac{\nu}{2}\mathbf{L}^T\mathbf{C}\Big)_+}{\alpha+\eta}. \tag{3}$$

Updating $\mathbf{U}$ *and* $\mathbf{V}$*:* The subproblem associated to $\mathbf{U}$ and $\mathbf{V}$ is

$$\min_{\mathbf{U},\mathbf{V}} \ \alpha\Big\|(\gamma\text{-diag}(\boldsymbol{\mathcal{X}} \times_1 \mathbf{U} \times_2 \mathbf{V}))^T - \mathbf{F}\Big\|_F^2 + \beta\big\|\mathbf{U}^T \odot \mathbf{V}^T\big\|_{2,1},$$

which is an unconstrained optimization problem.

Theorem 4.1. *The function*

$$\alpha\Big\|(\gamma\text{-diag}(\boldsymbol{\mathcal{X}} \times_1 \mathbf{U} \times_2 \mathbf{V}))^T - \mathbf{F}\Big\|_F^2 + \beta\big\|\mathbf{U}^T \odot \mathbf{V}^T\big\|_{2,1},$$

is convex with respect to $\mathbf{U}$ *(and with respect to* $\mathbf{V}$*).*

Proof. The proof can be found in the supplemental material. □

Although the subproblem associated to $\mathbf{U}$ (and to $\mathbf{V}$) is a convex optimization problem, it is impossible to derive a closed-form solution via first-order optimality conditions. Therefore, we adopt the simple gradient descent method. Denote the objective function of this subproblem as $\mathcal{J}_{\mathbf{U},\mathbf{V}}$, the gradients are then computed as

$$\frac{\partial\mathcal{J}_{\mathbf{U},\mathbf{V}}}{\partial\mathbf{U}} = 2\alpha\sum_{k=1}^{n}\mathrm{diag}(\mathbf{e}_{:k})\mathbf{V}\mathbf{X}^{(k)^T} + \beta\big(\mathbf{V}^{\circledast 2}\mathbf{Q}\big)\circledast\mathbf{U}, \tag{4}$$

$$\frac{\partial\mathcal{J}_{\mathbf{U},\mathbf{V}}}{\partial\mathbf{V}} = 2\alpha\sum_{k=1}^{n}\mathrm{diag}(\mathbf{e}_{:k})\mathbf{U}\mathbf{X}^{(k)} + \beta(\mathbf{U}^{\circledast 2}\mathbf{Q}^T)\circledast\mathbf{V}, \tag{5}$$

where $\mathbf{E} = \gamma\text{-diag}(\boldsymbol{\mathcal{X}} \times_1 \mathbf{U} \times_2 \mathbf{V}) - \mathbf{F}^T$, $\mathbf{V}^{\circledast 2} = \mathbf{V}\circledast\mathbf{V}$, and $\mathbf{Q}$ is the element-wise reciprocal of $\mathrm{vec}^{-1}_{n_2,n_1}$ $(\sqrt{((\mathbf{U}^T \odot \mathbf{V}^T)\circledast(\mathbf{U}^T \odot \mathbf{V}^T))\mathbf{1}_c})$. We then optimize $\mathbf{U}$ and $\mathbf{V}$ alternatively via the gradient descent method.

Summary: Based on the discussion above, we summarize the detailed alternating updating algorithm in Algorithm 1. The algorithm first constructs the graph Laplacian and randomly initializes all factor matrices (Lines 1-2). Afterwards, it alternately updates $\mathbf{A}$ and $\mathbf{B}$ by multiplicative rules derived from the classical NMF method (Lines 4-5), updates $\mathbf{C}$ and $\mathbf{F}$ analytically (Lines 6-7), and updates $\mathbf{U}$ and $\mathbf{V}$ by the gradient descent method (Lines 8-13). These steps are performed iteratively until some stopping criteria are met. We can then select features according to the ℓ_2-norms of rows of $\mathbf{U}^T \odot \mathbf{V}^T$.

Algorithm 1. The optimization algorithm for CPUFS

Input: Data tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1\times n_2\times n}$, parameters η, ν, α, β, learning rate θ and number of selected features p.

Output: The p selected features.

```
1:  Construct the weighted $k$-nearest neighbor graph and compute $\mathbf{L}$ accordingly, as in Section 3.3;
2:  Set $t \leftarrow 0$, initialize $\mathbf{A}_t$, $\mathbf{B}_t$, $\mathbf{C}_t$, $\mathbf{F}_t$, $\mathbf{U}_t$, $\mathbf{V}_t$ randomly;
3:  while $t < \Phi_1$ and not converged do
4:      Update $\mathbf{A}_{t+1} \leftarrow \mathbf{A}_t\circledast\frac{\mathbf{X}_{(1)}(\mathbf{C}_t\odot\mathbf{B}_t)}{\mathbf{A}_t(\mathbf{C}_t\odot\mathbf{B}_t)^T(\mathbf{C}_t\odot\mathbf{B}_t)}$;
5:      Update $\mathbf{B}_{t+1} \leftarrow \mathbf{B}_t\circledast\frac{\mathbf{X}_{(2)}(\mathbf{C}_t\odot\mathbf{A}_{t+1})}{\mathbf{B}_t(\mathbf{C}_t\odot\mathbf{A}_{t+1})^T(\mathbf{C}_t\odot\mathbf{A}_{t+1})}$;
6:      Update $\mathbf{C}_{t+1} \leftarrow \mathbf{V}_{\mathbf{C}}\mathbf{I}_{n,c}\mathbf{U}_{\mathbf{C}}^T$, where $2(\mathbf{B}_{t+1}\odot\mathbf{A}_{t+1})^T\mathbf{X}_{(3)}^T - \nu\mathbf{F}_t^T\mathbf{L}^T + 2\eta\mathbf{F}_t^T = \mathbf{U}_{\mathbf{C}}\mathbf{\Sigma}_{\mathbf{C}}\mathbf{V}_{\mathbf{C}}^T$;
7:      Update $\mathbf{F}_{t+1} \leftarrow \frac{1}{\alpha+\eta}(\alpha(\gamma\text{-diag}(\boldsymbol{\mathcal{X}}\times_1\mathbf{U}_t\times_2\mathbf{V}_t))^T + \eta\mathbf{C}_{t+1} - \frac{\nu}{2}\mathbf{L}^T\mathbf{C}_{t+1})_+$;
8:      Set $\tau \leftarrow 0$, and set $\mathbf{U}_\tau \leftarrow \mathbf{U}_t$, $\mathbf{V}_\tau \leftarrow \mathbf{V}_t$;
9:      while $\tau < \Phi_2$ and not converged do
10:         Update $\mathbf{U}_{\tau+1} \leftarrow \mathbf{U}_\tau - \theta\frac{\partial\mathcal{J}_{\mathbf{U},\mathbf{V}}}{\partial\mathbf{U}}|_{\mathbf{U}=\mathbf{U}_\tau,\mathbf{V}=\mathbf{V}_\tau,\mathbf{F}=\mathbf{F}_{t+1}}$ and $\mathbf{V}_{\tau+1} \leftarrow \mathbf{V}_\tau - \theta\frac{\partial\mathcal{J}_{\mathbf{U},\mathbf{V}}}{\partial\mathbf{V}}|_{\mathbf{U}=\mathbf{U}_{\tau+1},\mathbf{V}=\mathbf{V}_\tau,\mathbf{F}=\mathbf{F}_{t+1}}$ in order, with $\frac{\partial\mathcal{J}_{\mathbf{U},\mathbf{V}}}{\partial\mathbf{U}}$, $\frac{\partial\mathcal{J}_{\mathbf{U},\mathbf{V}}}{\partial\mathbf{V}}$ computed via (4) and (5);
11:         $\tau \leftarrow \tau + 1$;
12:     end while
13:     Update $\mathbf{U}_{t+1} \leftarrow \mathbf{U}_\tau$ and $\mathbf{V}_{t+1} \leftarrow \mathbf{V}_\tau$;
14:     $t \leftarrow t + 1$;
15: end while
16: return the $p$ selected features corresponding to the largest $p$ rows of $\mathbf{U}_t^T \odot \mathbf{V}_t^T$ in terms of the $\ell_2$-norm.
```

4.4 Convergence Analysis

The convergence of Algorithm 1 can be theoretically guaranteed. To show this, we denote the objective function of (2) to be $\mathcal{J}(\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{F},\mathbf{U},\mathbf{V})$.

Theorem 4.2. *Given a small enough learning rate* θ, $\mathcal{J}(\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{F},\mathbf{U},\mathbf{V})$ *is nonincreasing in each iteration of Algorithm 1 and converges to a local minimum.*

Proof. The proof consists of four parts as follows.

1) $\mathcal{J}(\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{F},\mathbf{U},\mathbf{V})$ *is bounded below:* We first show that when $\mathbf{C}^T\mathbf{C} = \mathbf{I}_c$, the term $\nu\mathrm{Tr}(\mathbf{C}^T\mathbf{L}\mathbf{F}) + \eta\|\mathbf{C}-\mathbf{F}\|_F^2$ is bounded below. By expanding the squared norm and then completing the square for $\mathbf{F}$, this term is equal to

$$\eta\left\|\mathbf{F} - \left(\mathbf{C} - \frac{\nu}{2\eta}\mathbf{L}\mathbf{C}\right)\right\|_F^2 + \nu\mathrm{Tr}\big(\mathbf{C}^T\mathbf{L}\mathbf{C}\big) - \frac{\nu^2}{4\eta}\mathrm{Tr}\big(\mathbf{C}^T\mathbf{L}^2\mathbf{C}\big).$$

Note that both $\mathbf{L}$ and $\mathbf{L}^2$ are positive semidefinite. Since $\mathbf{C}^T\mathbf{C} = \mathbf{I}_c$, the columns of $\mathbf{C}$ are orthonormal vectors. This leads to $\mathrm{Tr}(\mathbf{C}^T\mathbf{L}\mathbf{C}) \geq 0$ and $\mathrm{Tr}(\mathbf{C}^T\mathbf{L}^2\mathbf{C}) \leq \sum_{i=1}^n \lambda_i^2$ where λ_i is the ith largest eigenvalue of $\mathbf{L}$. Thus, we have

$$\nu\mathrm{Tr}\big(\mathbf{C}^T\mathbf{L}\mathbf{F}\big) + \eta\|\mathbf{C}-\mathbf{F}\|_F^2 \geq -\frac{\nu^2}{4\eta}\sum_{i=1}^n \lambda_i^2.$$

Therefore, $\mathcal{J}(\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{F},\mathbf{U},\mathbf{V})$ is bounded below by $-\frac{\nu^2}{4\eta}\sum_{i=1}^n \lambda_i^2$ by noticing that all the other terms in $\mathcal{J}(\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{F},\mathbf{U},\mathbf{V})$ are obviously nonnegative.

2) *Updating* $\mathbf{A}$ *and* $\mathbf{B}$*:* Our proposed multiplicative updating rules for $\mathbf{A}$ and $\mathbf{B}$ are directly derived from NMF counterparts and are guaranteed to decrease the objective functions of respective subproblems. Therefore, one has

$$\mathcal{J}(\mathbf{A}_{t+1}, \mathbf{B}_{t+1}, \mathbf{C}_t, \mathbf{F}_t, \mathbf{U}_t, \mathbf{V}_t) \le \mathcal{J}(\mathbf{A}_t, \mathbf{B}_t, \mathbf{C}_t, \mathbf{F}_t, \mathbf{U}_t, \mathbf{V}_t).$$

3) *Updating* $\mathbf{C}$ *and* $\mathbf{F}$*:* Since both $\mathbf{C}$ and $\mathbf{F}$ are updated by closed-form optimal solutions of respective subproblems, one has

$$\begin{aligned} &\mathcal{J}(\mathbf{A}_{t+1}, \mathbf{B}_{t+1}, \mathbf{C}_{t+1}, \mathbf{F}_{t+1}, \mathbf{U}_t, \mathbf{V}_t) \\ &\le \mathcal{J}(\mathbf{A}_{t+1}, \mathbf{B}_{t+1}, \mathbf{C}_t, \mathbf{F}_t, \mathbf{U}_t, \mathbf{V}_t). \end{aligned}$$

4) *Updating* $\mathbf{U}$ *and* $\mathbf{V}$*:* Since $\mathbf{U}$ and $\mathbf{V}$ are updated via the gradient descent method, given a small enough learning rate θ, it is guaranteed that the objective function is decreasing in every gradient descent iteration $\tau \to \tau + 1$ for $\tau = 0, 1, \ldots, \Phi_2 - 1$. Thus, we have

$$\begin{aligned} &\mathcal{J}(\mathbf{A}_{t+1}, \mathbf{B}_{t+1}, \mathbf{C}_{t+1}, \mathbf{F}_{t+1}, \mathbf{U}_{t+1}, \mathbf{V}_{t+1}) \\ &\le \mathcal{J}(\mathbf{A}_{t+1}, \mathbf{B}_{t+1}, \mathbf{C}_{t+1}, \mathbf{F}_{t+1}, \mathbf{U}_t, \mathbf{V}_t). \end{aligned}$$

By combining the last three inequalities, we conclude that

$$\begin{aligned} &\mathcal{J}(\mathbf{A}_{t+1}, \mathbf{B}_{t+1}, \mathbf{C}_{t+1}, \mathbf{F}_{t+1}, \mathbf{U}_{t+1}, \mathbf{V}_{t+1}) \\ &\le \mathcal{J}(\mathbf{A}_t, \mathbf{B}_t, \mathbf{C}_t, \mathbf{F}_t, \mathbf{U}_t, \mathbf{V}_t). \end{aligned}$$

Therefore, the objective values computed by Algorithm 1 converges to a local minimum as the iterations are nonincreasing and bounded below. □

4.5 Computational Complexity Analysis

We now analyze the asymptotic computational complexity of Algorithm 1 step by step as follows. As in reality, we assume that $c \ll n$.

1) *Updating* $\mathbf{A}$*:* One must compute $\mathbf{C} \odot \mathbf{B}$ first whose computational complexity is $\mathcal{O}(n_2nc)$. Besides, computing the multiplicative updating rule costs $\mathcal{O}(n_1n_2nc + n_2nc + n_2nc^2 + n_1c^2 + n_1c + n_1c)$ time as we compute $(\mathbf{C} \odot \mathbf{B})^T(\mathbf{C} \odot \mathbf{B})$ first instead of $\mathbf{A}(\mathbf{C} \odot \mathbf{B})^T$. Thus, the total time complexity of updating $\mathbf{A}$ is $\mathcal{O}(n_1n_2nc + n_2nc^2)$.
2) *Updating* $\mathbf{B}$*:* Similar to the above, the time complexity of updating $\mathbf{B}$ is $\mathcal{O}(n_1n_2nc + n_1nc^2)$.
3) *Updating* $\mathbf{C}$*:* One must compute $\mathbf{Q} = 2(\mathbf{B} \odot \mathbf{A})^T\mathbf{X}_{(3)}^T - \nu\mathbf{F}^T\mathbf{L}^T + 2\eta\mathbf{F}^T$ first whose computational complexity is $\mathcal{O}(n^2c + n_1n_2nc)$. Besides, conducting singular value decomposition by bi-diagonalization and QR algorithm on that matrix costs $\mathcal{O}(n^2c)$ time. Finally, computing $\mathbf{V}_\mathbf{C}\mathbf{I}_{n,c}\mathbf{U}_\mathbf{C}^T$ needs $\mathcal{O}(n^2c)$ time. Thus, the total time complexity of updating $\mathbf{C}$ is $\mathcal{O}(n_1n_2nc + n^2c)$.
4) *Updating* $\mathbf{F}$*:* Computing items in the parenthesis in (3) requires $\mathcal{O}(n_1n_2nc + n_2nc^2 + n^2c)$ time as we compute $\mathcal{X} \times_1 \mathbf{U}$ first instead of $\mathcal{X} \times_2 \mathbf{V}$. Picking the nonnegative entries on those items costs $\mathcal{O}(nc)$ time. Thus, the total time complexity of updating $\mathbf{F}$ is $\mathcal{O}(n_1n_2nc + n_2nc^2 + n^2c)$.
5) *Updating* $\mathbf{U}$ *and* $\mathbf{V}$*:* To compute $\frac{\partial \mathcal{J}_{\mathbf{U},\mathbf{V}}}{\partial \mathbf{U}}$ and $\frac{\partial \mathcal{J}_{\mathbf{U},\mathbf{V}}}{\partial \mathbf{V}}$, one needs to compute $\mathbf{E}$ and $\mathbf{Q}$ in advance, whose computational complexities are $\mathcal{O}(n_1n_2nc + n_2nc^2)$ and $\mathcal{O}(n_1n_2c)$, respectively. Besides, since $\mathrm{diag}(\mathbf{e}_k)$ is a diagonal matrix, computing $\frac{\partial \mathcal{J}_{\mathbf{U},\mathbf{V}}}{\partial \mathbf{U}}$ and $\frac{\partial \mathcal{J}_{\mathbf{U},\mathbf{V}}}{\partial \mathbf{V}}$ costs the same $\mathcal{O}(n_1n_2nc)$ time. Thus, the total time complexity for one iteration of the gradient descent loop is $\mathcal{O}(n_1n_2nc + n_2nc^2)$.

To summarize, as the maximum numbers of iterations Φ_1 and Φ_2 are all constants, the total asymptotic complexity for each iteration of Algorithm 1 is $\mathcal{O}(n_1nc^2 + n_2nc^2 + n_1n_2nc + n^2c)$, where n_i is the number of mode-i features for $i = 1, 2$, n is the number of samples, and c is the number of latent classes.

Discussions: Compared with other relevant unsupervised feature selection methods, such as NDFS whose computational complexity is $\mathcal{O}((n_1n_2)^3 + n^2c)$, CPUFS bypasses the curse of dimensionality from the algorithmic aspect. The computational time for our method scales linearly in the number of features n_1n_2.

4.6 CPUFSnn: A Variation of the CPUFS Model

The model CPUFS can be modified to incorporate the nonnegativity of $\mathbf{U}$ and $\mathbf{V}$ if required. In our CPUFS setting, the pseudo cluster indicator matrix $\mathbf{C}$ and the data tensor $\mathcal{X}$ are both nonnegative. Therefore, the linear classifier should be constrained to be nonnegative as well. This would benefit the pseudo label fitting process because nonnegative $\mathbf{U}$ and $\mathbf{V}$ only allow feature additions that retain the semantic meanings of the features but prohibit subtractions or other operations that may distort the semantic meanings. In order to validate this hypothesis, we derive a modified CPUFS model, named CPUFSnn as a comparison, where the nonnegative constraint is imposed on both $\mathbf{U}$ and $\mathbf{V}$. Specifically, the CPUFSnn model is

$$\begin{aligned} \min_{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{U},\mathbf{V},\mathbf{Y}} \quad & \left\|\mathcal{X} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]\right\|_F^2 + \nu\mathrm{Tr}\big(\mathbf{C}^\top\mathbf{L}\mathbf{C}\big) \\ & + \alpha\left\|(\gamma\text{-diag}(\mathcal{X} \times_1 \mathbf{U} \times_2 \mathbf{V}))^\top - \mathbf{C}\right\|_F^2 + \beta\left\|\mathbf{U}^\top \odot \mathbf{V}^\top\right\|_{2,1} \\ \text{s.t.} \quad & \mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{U}, \mathbf{V} \ge 0, \mathbf{C} = \mathbf{Y}\big(\mathbf{Y}^\top\mathbf{Y}\big)^{-\frac{1}{2}}, \mathbf{Y} \in \{0, 1\}^{n \times c}. \end{aligned}$$

To solve the CPUFSnn model, we apply the same relaxation strategy as that for CPUFS in Section 4.3.1 and subsequently the same learning procedure as that for CPUFS in Algorithm 1 except for the step of updating $\mathbf{U}$ and $\mathbf{V}$. In that step, the projected gradient descent method is adopted to replace the gradient descent method. We will compare the performance of both CPUFS and CPUFSnn in the experiments.

5 Experiments

5.1 Experimental Settings

5.1.1 Experimental Environment and Datasets

All the experiments are implemented in MATLAB R2020b and run on a Ubuntu server with 3.70-GHz i9-10900K CPU and 128GB main memory. We perform the experiments on ten real-world benchmark datasets, including two object classification datasets (FashionMNIST [42] and COIL20 [43]), five

TABLE 2
Detailed Statistics of the Ten Datasets

Dataset	# of Samples	Feature Size	# of Classes	Range of Selected Features
FashionMNIST	1,000	28×28	10	$[50, 100, 150, \ldots, 300]$
COIL20	1,440	32×32	20	$[50, 100, 150, \ldots, 300]$
ORL	400	32×32	40	$[50, 100, 150, \ldots, 300]$
UMIST	575	23×28	20	$[50, 100, 150, \ldots, 300]$
Pixraw10P	100	100×100	10	$[50, 100, 150, \ldots, 300]$
Orlraws10P	100	92×112	10	$[50, 100, 150, \ldots, 300]$
BreastMNIST	200	28×28	2	$[50, 100, 150, \ldots, 300]$
OCTMNIST	400	28×28	4	$[50, 100, 150, \ldots, 300]$
OrganSMNIST	220	28×28	11	$[50, 100, 150, \ldots, 300]$
JAFFE	213	64×64	10	$[50, 100, 150, \ldots, 300]$

face classification datasets (ORL [16], UMIST [44], Pixraw10P[2], Orlraws10P[3] and JAFFE [45]), and three medical image classification datasets (BreastMNIST [46], OCTMNIST [46] and OrganSMNIST [46]). For all the datasets, we normalize the value of each feature to the range of 0 and 1. Since the scales of BreastMNIST, OCTMNIST and OrganSMNIST are too large, we only select 100, 100 and 10 samples from each class of these three datasets, respectively. We summarize details of these datasets in Table 2.

5.1.2 Comparative Methods

To validate the effectiveness of our proposed CPUFS and CPUFSnn[3], we compare them with several state-of-the-art unsupervised feature selection methods[4], including AllFea (a baseline method that simply selects all the original features), LapScore[5] [15], MCFS[6] [16], UDFS[6] [17], SOCFS[7] [22], SOGFS [24], RUFS[8] [20], RSFS[9] [21], JELSR[10] [18] and CDLFS[11] [25]. The detailed descriptions for these methods can be found in Section 2.1. Codes for SOGFS are implemented by ourselves and codes for all other comparative methods are provided by their original authors as footnoted.

5.1.3 Parameter Settings

To fairly compare different unsupervised feature selection methods, we tune all parameters for all methods by a grid-search strategy in the range of $\{10^{-2}, 10^{-1}, 1, 10, 10^{2}\}$. For CPUFS, we set $\eta = 10^5$ across all the datasets, while for CPUFSnn, we set $\eta = 10^5$ for all but the BreastMNIST, OCTMNIST and OrganSMNIST datasets where we set $\eta = 10^4$ consistently[12]. For those methods which utilize the weighted k-nearest neighbor graph, we set the number of neighbors $k = 5$ and the Gaussian kernel width $\sigma = 1$. For projection-based methods, we set the dimensionality of the projected subspace as the ground-truth number of classes c, and for clustering-based methods, we also set the number of latent classes as c. Besides, we set $\Phi_1 = 500$ and $\Phi_2 = 2$ and define the stopping criterion of Algorithm 1 as reaching the maximum number of iterations Φ_1.

5.1.4 Evaluation Methodology and Evaluation Metrics

After features have been selected, we evaluate the performance of feature selection by conducting k-means clustering on the selected features and adopt two widely used metrics *accuracy (ACC)* and *normalized mutual information (NMI)* to quantitatively describe the clustering performance [13]. The larger the metric values, the better the clustering performance, and accordingly the higher the quality of the selected features. Since the results of the k-means clustering partly depend on the initialization, we adopt the following strategy to alleviate the stochastic effects inherently existing in the evaluation system. Specifically, for every group of the selected features, we repeat the k-means clustering 20 times with random initialization and then record the average result with the obtained standard deviation. For each method, its clustering results from the optimal parameters are reported.

5.2 Experiment 1: Clustering on the Selected Features

In this section, we report and analyze the performance of different unsupervised feature selection methods in terms of the clustering performance on the selected features. The results of clustering performance are shown in Fig. 3 and Fig. 4 where the shaded area represents the interval $[\mu - 0.2\sigma, \mu + 0.2\sigma]$,[13] the black curve with bigger diamond markers represents CPUFS, and the dark blue curve with smaller diamond markers represents CPUFSnn. Based on Fig. 3 and Fig. 4, we have the following observations.

1) *Feature selection improves data quality*. Compared with the baseline method ALLfea, the k-means clustering performance based on features selected by most methods has been enhanced. This shows that the feature selection can indeed filter out redundant and noisy features and is thus very important.
2) *CPUFS and CPUFSnn outperform the state-of-the-art methods*. In terms of the maximum attainable NMI, at least one of our proposed methods consistently get the best performance on all the ten datasets. While in terms of the maximum attainable ACC, although our proposed methods are not always the best, they still outperform other comparative methods remarkably on all but the FashionMNIST and UMIST datasets. This has fully demonstrated the effectiveness of CPUFS and CPUFSnn. We believe that it mainly attributes to the consideration of incorporating the multi-dimensional data structure.
3) *Both CPUFS and CPUFSnn are competent while their applicability may vary across different datasets*. From Fig. 3 and Fig. 4, it is hard to determine whether CPUFS or CPUFSnn is better. For example, on the

[2] See https://jundongl.github.io/scikit-feature/datasets.html.
[3] Codes are publicly available at https://github.com/Kwan1997/CPUFS.
[4] We do not compare with the tensor-based unsupervised feature selection method GRLTR [35] as it is an embedded method and is beyond the scope of this paper.
[5] See https://github.com/ZJULearning/MatlabFunc/.
[6] See http://www.cs.cmu.edu/~yiyang/.
[7] See https://sites.google.com/site/dyhan0920/.
[8] See https://sites.google.com/site/qianmingjie/.
[9] See https://github.com/LeiShiCS/RSFS/.
[10] See https://sites.google.com/site/houchenping/.
[11] See https://github.com/AISKYEYE-TJU/CDLFS-AAAI2016/.
[12] Our parameter for η are tested to be adequate enough to diminish the discrepancy between $\mathbf{C}$ and $\mathbf{F}$ in both CPUFS and CPUFSnn but we do not report this part of experimental results due to the page limit.
[13] Here, μ and σ represent the mean and the standard deviation of the 20 k-means clustering trials, respectively.

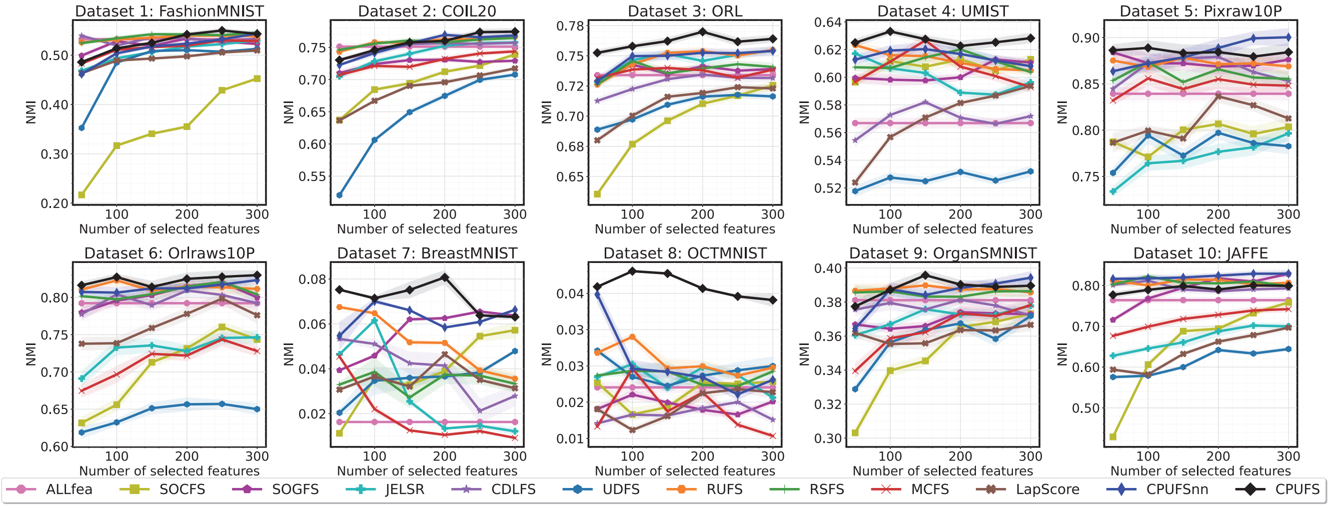


Fig. 3. NMI curves of different unsupervised feature selection methods on the ten datasets.

JAFFE dataset, CPUFSnn yields better performance than CPUFS, while on the ORL dataset, CPUFSnn is inferior to CPUFS. This suggests that the nonnegativity of the linear classifier may not always be conducive for feature selection, but in some cases, it does increase the quality of the selected features.

5.3 Experiment 2: Parameter Sensitivity Analysis

In this section, we analyze the impact of the parameters ν, α and β in CPUFS on the performance of feature selection. In particular, we alternately vary one of these parameters in the range of $\{10^{-2}, 10^{-1}, 1, 10, 10^{2}\}$ while fixing the others to 1 and record the feature selection performance (in terms of NMI) of CPUFS under all possible parameter combinations. For the interest of space, we only illustrate results on the FashionMNIST and UMIST datasets in Fig. 5. We have the following observations:

1) For the FashionMNIST dataset, the performance of CPUFS is relatively sensitive to these parameters when the number of selected features is low (say, 50 or 100) while it is relatively nonsensitive in other cases. Besides, the performance of CPUFS shows a positive correlation with respect to the number of selected features.
2) For the UMIST dataset, the performance of CPUFS is relatively nonsensitive to all these parameters and is relatively stable with respect to the number of selected features.

5.4 Experiment 3: Running Time Analysis

In this section, we analyze the running time of our proposed CPUFS method. As analyzed in Section 4.5, the computational complexity of CPUFS scales linearly in the number of features while some previous methods scale cubically. To validate our analysis, we run CPUFS, NDFS, UDFS and RSFS on the Pixraw10P and Orlraws10P datasets which have the largest numbers of features among all the datasets (and hence are more suitable to compare the running time). Specifically, we run the four methods on the two datasets with all tunable parameters fixed to 1 and then record their accumulative training time within 50 iterations. The experimental results are shown in Fig. 6(a) and Fig. 6(b). We observe that our CPUFS method takes significantly less

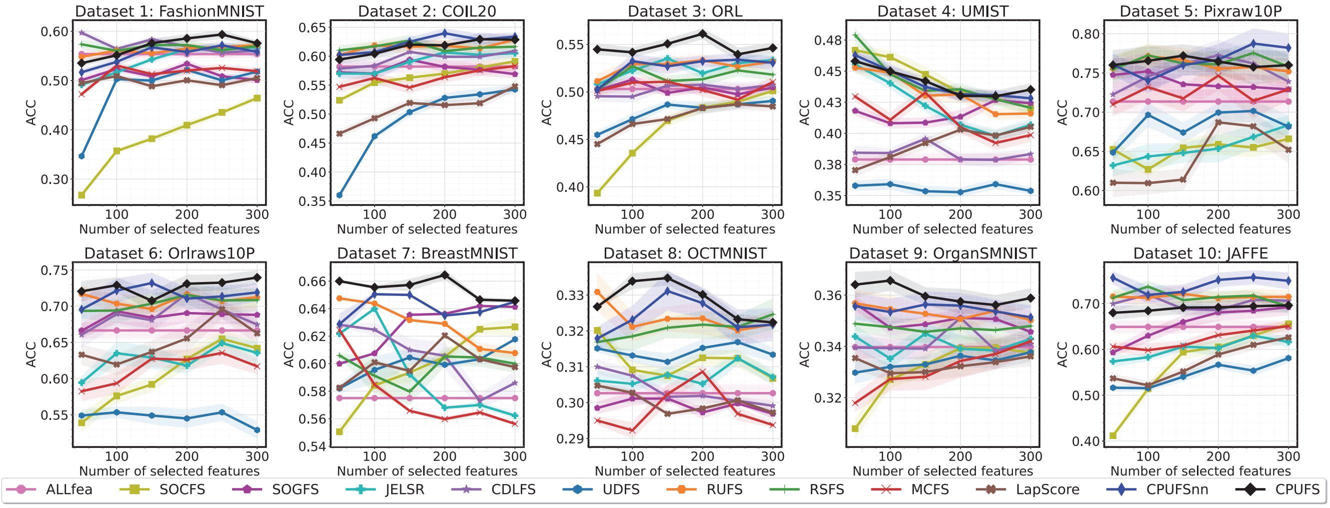
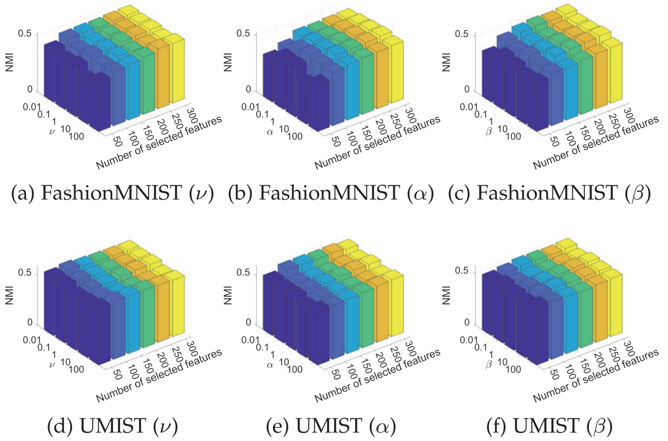


Fig. 4. ACC curves of different unsupervised feature selection methods on the ten datasets.



(a) FashionMNIST (ν) (b) FashionMNIST (α) (c) FashionMNIST (β)

(d) UMIST (ν) (e) UMIST (α) (f) UMIST (β)

Fig. 5. Parameter sensitivity analysis of CPUFS on the FashionMNIST and UMIST datasets.

training time than NDFS, UDFS and RSFS. It clearly supports our theoretical complexity analysis.

5.5 Experiment 4: Convergence Analysis

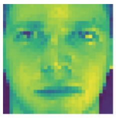
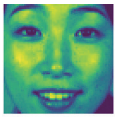
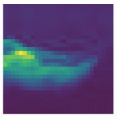
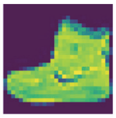
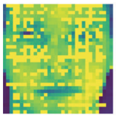
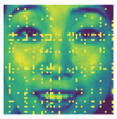
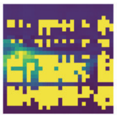
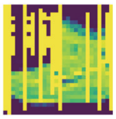
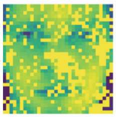
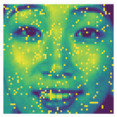
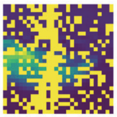
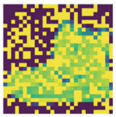
In this part, we empirically study the convergence speed of Algorithm 1. In particular, we run CPUFS on the FashionMNIST dataset with all parameters fixed to 1 and then record its objective function value within 105 iterations. The corresponding convergent curve is shown in Fig. 6(c). As observed, the objective function value decreases monotonically. More specifically, it drops rapidly at the very beginning and then decreases steadily to a stationary point. This validates the convergence theory Theorem 4.2. Moreover, it can be seen from Fig. 6(c) that the objective function decreases to a low level via Algorithm 1 within only dozens of iterations. This ensures the efficiency of the whole feature selection process.

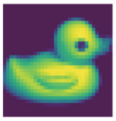
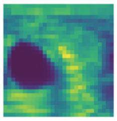
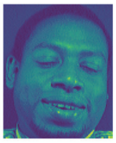
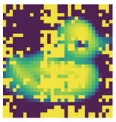
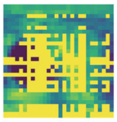
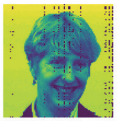
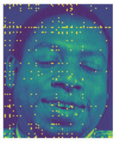
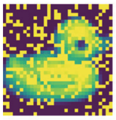
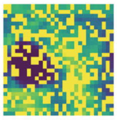
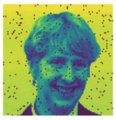
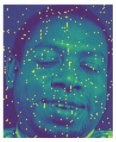
5.6 Experiment 5: Selected Feature Visualization

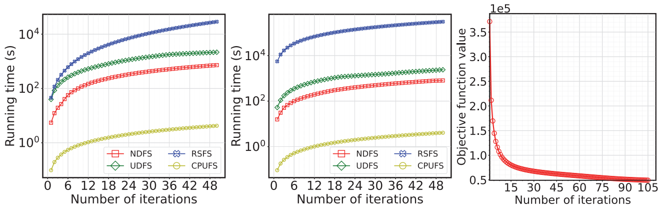
As discussed at the end of Section 4.1, in the CPUFS model, the importance weights of features from the same row or the same column of the original two-dimensional data are directly correlated. We now empirically show how this correlation appears in practice. In particular, for a specific dataset, we first retrieve 300 best performing features in terms of NMI (i.e., these 300 features are exactly the ones that performed the best in Fig. 3) and then mask these 300 selected features on a randomly sampled image from that dataset. Since our method is relatively most similar to RUFS, we also visualize the features selected by RUFS following the same routine. We illustrate experimental results on the ORL, JAFFE, OCTMNIST, FashionMNIST, COIL20, BreastMNIST, Pixraw10P and Orlraws10P datasets in Table 3. It can be seen that the features selected by our CPUFS method are spatially more correlated:

1) On the ORL dataset, the selected features tend to gather within a (smaller) rectangular sub-region;
2) On the JAFFE and Orlraws10P datasets, the grid-like characteristic of the selected features is obvious;
3) On the OCTMNIST and BreastMNIST datasets, the selected features are more likely to stick together;
4) On the FashionMNIST, COIL20 and Pixraw10P datasets, the vertical characteristic of the selected features is very clear.

TABLE 3
Visualization of the Selected Features

Method	ORL	JAFFE	OCTMNIST	FashionMNIST
Original				
CPUFS				
RUFS				

Method	COIL20	BreastMNIST	Pixraw10P	Orlraws10P
Original				
CPUFS				
RUFS				



(a) Pixraw10P (b) Orlraws10P (c) FashionMNIST

Fig. 6. Running time and convergence analyses of CPUFS on the Pixraw10P, Orlraws10P and FashionMNIST datasets.

However, selected features from RUFS show few such effects. This phenomenon fully demonstrates the effectiveness of a tensor-based method. We believe that the improved performance by CPUFS mainly attributes to the better structure of its selected features.

6 Conclusion

In this paper, we proposed a novel unsupervised feature selection method, namely, nonnegative tensor CP decomposition based unsupervised feature selection (CPUFS). Distinct from existing unsupervised feature selection methods, CPUFS takes into account the multi-dimensional structural information of data, which is neglected by most existing methods. To solve the CPUFS model, we proposed an efficient iterative optimization algorithm and established its convergence. Moreover, the proposed optimization algorithm scales linearly in the number of features. A variation of the CPUFS model, called CPUFSnn, has also been proposed and studied. Extensive experiments have been conducted on ten real-world benchmark datasets. The experimental results demonstrated that our proposed CPUFS and CPUFSnn methods clearly outperform the state-of-the-arts. Besides, we also tested the parameter sensitivity and empirically analyzed the running time and the convergent speed of CPUFS. It is worth mentioning that our proposed methods can be easily extended to higher-order tensors.

For future work, one possible direction is how to endow CPUFS with the ability to handle missing data. In some real-world scenarios, the data might be incomplete, due to, e.g., improper data collection or intentional (malicious) corruption. However, directly applying the proposed CPUFS method to these scenarios is not favorable as CPUFS does not admit any built-in mechanism for handling missing data and may lead to a degraded feature selection performance. A possible approach is to adapt missing entry estimation mechanisms for CPUFS. As an example, one can adopt the strategy introduced in [34] to enhance CPUFS, i.e., introducing an additional optimization variable $\mathcal{Y}$ to replace $\mathcal{X}$ in the objective function and imposing an equality constraint that enforces the entries of $\mathcal{X}$ and $\mathcal{Y}$ to be identical in the positions where $\mathcal{X}$ is complete.

Acknowledgments

The authors would like to thank the anonymous referees for their valuable comments.

References

[1] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[2] R. O. Duda and P. E. Hart, *Pattern Classification*. Hoboken, NJ, USA: Wiley, 2006.

[3] Y. Sun, S. Todorovic, and S. Goodison, "Local-learning-based feature selection for high-dimensional data analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1610–1626, Sep. 2010.

[4] X. Wu, X. Xu, J. Liu, H. Wang, B. Hu, and F. Nie, "Supervised feature selection with orthogonal regression and feature weighting," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 5, pp. 1831–1838, May 2021.

[5] X. Li, Y. Zhang, and R. Zhang, "Semisupervised feature selection via generalized uncorrelated constraint and manifold embedding," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Apr. 02, 2021, doi: 10.1109/TNNLS.2021.3069038.

[6] X. Chen, R. Chen, Q. Wu, F. Nie, M. Yang, and R. Mao, "Semisupervised feature selection via structured manifold learning," *IEEE Trans. Cybern.*, early access, Feb. 26, 2021, doi: 10.1109/TCYB.2021.3052847.

[7] F. Nie, X. Dong, L. Tian, R. Wang, and X. Li, "Unsupervised feature selection with constrained $\ell_{2,0}$-norm and optimized graph," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Dec. 25, 2020, doi: 10.1109/TNNLS.2020.3043362.

[8] Z. Li, F. Nie, J. Bian, D. Wu, and X. Li, "Sparse pca via $\ell_{2,p}$-norm regularization for unsupervised feature selection," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Oct. 19, 2021, doi: 10.1109/TPAMI.2021.3121329.

[9] X. Lin, J. Guan, B. Chen, and Y. Zeng, "Unsupervised feature selection via orthogonal basis clustering and local structure preserving," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jan. 08, 2021, doi: 10.1109/TNNLS.2021.3083763.

[10] S. Solorio-Fernández, J. A. Carrasco-Ochoa, and J. F. Martínez-Trinidad, "A review of unsupervised feature selection methods," *Artif. Intell. Rev.*, vol. 53, no. 2, pp. 907–948, 2020.

[11] Y. Zhao *et al.*, "Four-dimensional modeling of fmri data via spatio-temporal convolutional neural networks," *IEEE Trans. Cogn. Devel. Syst.*, vol. 12, no. 3, pp. 451–460, Sep. 2020.

[12] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Rev.*, vol. 51, no. 3, pp. 455–500, 2009.

[13] D. Cai, X. He, J. Han, and T. S. Huang, "Graph regularized nonnegative matrix factorization for data representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1548–1560, Aug. 2011.

[14] J. D. Carroll and J.-J. Chang, "Analysis of individual differences in multidimensional scaling via an n-way generalization of 'Eckart-Young' decomposition," *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970.

[15] X. He, D. Cai, and P. Niyogi, "Laplacian score for feature selection," in *Proc. 19th Int. Conf. Neural Inf. Process. Syst.*, 2006, pp. 507–514.

[16] D. Cai, C. Zhang, and X. He, "Unsupervised feature selection for multi-cluster data," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2010, pp. 333–342.

[17] Y. Yang, H. T. Shen, Z. Ma, Z. Huang, and X. Zhou, "$\ell_{2,1}$-norm regularized discriminative feature selection for unsupervised learning," in *Proc. 21st Int. Joint Conf. Artif. Intell.*, 2011, pp. 1589–1594.

[18] C. Hou, F. Nie, D. Yi, and Y. Wu, "Feature selection via joint embedding learning and sparse regression," in *Proc. 22nd Int. Joint Conf. Artif. Intell.*, 2011, pp. 1324–1329.

[19] Z. Li, Y. Yang, J. Liu, X. Zhou, and H. Lu, "Unsupervised feature selection using nonnegative spectral analysis," in *Proc. 26th AAAI Conf. Artif. Intell.*, 2012, pp. 1026–1032.

[20] M. Qian and C. Zhai, "Robust unsupervised feature selection," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, 2013, pp. 1621–1627.

[21] L. Shi, L. Du, and Y.-D. Shen, "Robust spectral learning for unsupervised feature selection," in *Proc. 14th IEEE Int. Conf. Data Mining*, 2014, pp. 977–982.

[22] D. Han and J. Kim, "Unsupervised simultaneous orthogonal basis clustering feature selection," in *Proc. 28th IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 5016–5023.

[23] L. Du and Y.-D. Shen, "Unsupervised feature selection with adaptive structure learning," in *Proc. 21st ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2015, pp. 209–218.

[24] F. Nie, W. Zhu, and X. Li, "Unsupervised feature selection with structured graph optimization," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 1302–1308.

[25] P. Zhu, Q. Hu, C. Zhang, and W. Zuo, "Coupled dictionary learning for unsupervised feature selection," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 2422–2428.

[26] X. Li, H. Zhang, R. Zhang, Y. Liu, and F. Nie, "Generalized uncorrelated regression with adaptive graph for unsupervised feature selection," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 5, pp. 1587–1595, May 2019.

[27] J. Guo and W. Zhu, "Dependence guided unsupervised feature selection," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 2232–2239.

[28] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics Intell. Lab. Syst.*, vol. 2, no. 1-3, pp. 37–52, 1987.

[29] H. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos, "MPCA: Multilinear principal component analysis of tensor objects," *IEEE Trans. Neural Netw.*, vol. 19, no. 1, pp. 18–39, Jan. 2008.

[30] X. Li, M. K. Ng, G. Cong, Y. Ye, and Q. Wu, "Mr-ntd: Manifold regularization nonnegative tucker decomposition for tensor data dimension reduction and representation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 8, pp. 1787–1800, Aug. 2017.

[31] C. Lu, J. Feng, Y. Chen, W. Liu, Z. Lin, and S. Yan, "Tensor robust principal component analysis: Exact recovery of corrupted low-rank tensors via convex optimization," in *Proc. 29th IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 5249–5257.

[32] E. J. Candés, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?," *J. ACM*, vol. 58, no. 3, pp. 1–37, 2011.

[33] F. Ju, Y. Sun, J. Gao, Y. Hu, and B. Yin, "Vectorial dimension reduction for tensors based on bayesian inference," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 10, pp. 4579–4592, Oct. 2018.

[34] Q. Shi, Y.-M. Cheung, Q. Zhao, and H. Lu, "Feature extraction for incomplete data via low-rank tensor decomposition with feature regularization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 6, pp. 1803–1817, Jun. 2019.

[35] Y. Su, X. Bai, W. Li, P. Jing, J. Zhang, and J. Liu, "Graph regularized low-rank tensor representation for feature selection," *J. Vis. Commun. Image Representation*, vol. 56, pp. 234–244, 2018.

[36] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.

[37] T. G. Kolda, "Multilinear operators for higher-order decompositions." Sandia National Laboratories, Tech. Rep. SAND2006-2081, 2006.

[38] H. Huang, C. Ding, D. Luo, and T. Li, "Simultaneous tensor subspace selection and clustering: The equivalence of high order SVD and k-means clustering," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2008, pp. 327–335.

[39] B. W. Bader and T. G. Kolda, "Algorithm 862: MATLAB tensor classes for fast algorithm prototyping," *ACM Trans. Math. Softw.*, vol. 32, no. 4, pp. 635–653, 2006.

[40] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Proc. 14th Int. Conf. Neural Inf. Process. Syst.*, 2001, pp. 556–562.

[41] T. Viklands, "Algorithms for the weighted orthogonal procrustes problem and other least squares problems," Ph.D. dissertation, Dept. Comput. Sci., Umeå Univ., Umeå, Sweden, 2006.

[42] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*.

[43] S. A. Nene, S. K. Nayar, and H. Murase, "Columbia object image library (coil-20)" Columbia Univ., Tech. Rep. CUCS-006-96, 1996.

[44] D. B. Graham and N. M. Allinson, "Characterising virtual eigensignatures for general purpose face recognition" *Face Recognit., Theory Appl., NATO ASI Ser. F., Comput. Syst. Sci.*, vol. 163, pp. 446–456, 1998.

[45] M. J. Lyons, J. Budynek, and S. Akamatsu, "Automatic classification of single facial images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 12, pp. 1357–1362, Dec. 1999.

[46] J. Yang, R. Shi, and B. Ni, "Medmnist classification decathlon: A lightweight automl benchmark for medical image analysis," in *Proc. 18th IEEE Int. Symp. Biomed. Imag.*, 2021, pp. 191–195.

Bilian Chen received the PhD degree from The Chinese University of Hong Kong, Hong Kong, in 2012. She is currently an associate professor with Xiamen University, Xiamen, China. Her publications appear in *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Neural Networks and Learning Systems*, *SIAM Journal on Optimization*, and *Information Sciences*. Her research interests include machine learning, optimization theory, and recommendation system.

Jiewen Guan received the BEng degree from Zhejiang University of Technology, Hangzhou, China, in 2019. He is currently working toward the MEng degree with Xiamen University, Xiamen, China. His publications appear in *IEEE Transactions on Neural Networks and Learning Systems* and *IEEE Transactions on Knowledge and Data Engineering*. His research interests include the intersection between data mining and optimization.

Zhening Li is currently with the School of Mathematics and Physics and the Centre for Operational Research and Logistics in the University of Portsmouth, Portsmouth, U.K.