Pattern Recognition 105 (2020) 107375

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Unsupervised feature selection with adaptive multiple graph learning

Peng Zhou[a,b,*], Liang Du[c], Xuejun Li[a], Yi-Dong Shen[b], Yuhua Qian[c]

[a] School of Computer Science and Technology, Anhui University, Hefei 230601, China
[b] The State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
[c] School of Computer and Information Technology, Shanxi University, Taiyuan 030006, China

ARTICLE INFO

Article history:
Received 21 May 2019
Revised 31 March 2020
Accepted 12 April 2020
Available online 28 April 2020

Keywords:
Feature selection
Multiple graph learning
Consensus learning

ABSTRACT

Unsupervised feature selection methods try to select features which can well preserve the intrinsic structure of data. To represent such structure, conventional methods construct various graphs from data. In most cases, those different graphs often contain some consensus and complementary information. To make full use of such information, we construct multiple base graphs and learn an adaptive consensus graph from these base graphs for feature selection. In our method, we integrate the multiple graph learning and the feature selection into a unified framework, which can jointly characterize the structure of the data and select the features to preserve such structure. The underlying optimization problem is hard to solve, and we solve it via a block coordinate descent schema, whose convergence is guaranteed. The extensive experiments well demonstrate the effectiveness of our proposed framework.

1. Introduction

Feature selection is a fundamental problem in machine learning and has attracted considerable attention in the past decades [1–5]. In many real-world applications, the data contain a large number of features, which may cause the curse of dimensionality. Moreover, some features are often contaminated by noises, which may deteriorate the performance of machine learning methods. To address these problems, feature selection is applied to select a small number of informative and non-redundant features for the machine learning methods. Since feature selection usually leads to better learning performance, it has been successfully applied in many real applications such as text categorization [6,7], image processing [8], and bioinformatics [9]. According to the availability of the labels, these methods can be categorized into supervised [10,11], semi-supervised [12] and unsupervised algorithms [5,13]. Due to the absence of the label information, the unsupervised feature selection is a more challenging problem.

Since the label information is absent in unsupervised learning, feature selection method can only make use of the information of data itself, or equivalently speaking, the intrinsic structure of data [14–18]. Conventional methods often construct a graph from the data to represent such structure. According to the way of constructing the graph, the existing methods can be roughly categorized into two classes: (1) **using a pre-defined graph**, which often pre-defines a graph and selects features to preserve such graph structure [19–21]; (2) **leaning an adaptive graph**, which learns an adaptive graph simultaneously in the process of feature selection [5,17,22].

Note that both the two classes only use one single graph (either a pre-defined graph or an adaptive graph) to represent the structure of data. However, in real applications, the structure may be too complex to be captured by one single graph. Given a data set, various graphs can be constructed based on different distance metrics, such as Euclidean distance and cosine similarity. In most cases, these different graphs contain some consensus and complementary information. Those conventional methods which only use one single graph may ignore such useful information. Moreover, some data are naturally performed in multiple graph structures. For example, relationships of research papers contain several graphs such as co-author graph and citation graph. When handling these data, the aforementioned methods may fail to fully utilize the given graphs.

To address these problems, in this paper, we characterize the intrinsic structure with adaptive multiple graph learning. Multiple graph learning tries to learn a consensus graph from multiple base graphs. For example, Nie et al. [23] proposed a parameter-free multiple graph learning method which learned the weight of each graph automatically; Zhan et al. [24] learned a consensus graph with minimizing disagreement between different views and constraining the rank of the Laplacian matrix. In our framework, firstly, if the data contains multiple graphs, we directly use them

* Corresponding author.
E-mail addresses: zhoupeng@ahu.edu.cn (P. Zhou), xjli@ahu.edu.cn (X. Li), ydshen@ios.ac.cn (Y.-D. Shen), jinchengqyh@sxu.edu.cn (Y. Qian).

https://doi.org/10.1016/j.patcog.2020.107375
0031-3203/

as base graphs; if otherwise, we construct some pre-defined graphs from data. Then we learn the consensus graph from these base graphs simultaneously in the process of feature selection. On the one hand, we use the result of feature selection to guide the multiple graph learning, and on the other hand, we apply the learned graph to select the informative features. When learning the consensus graph, since the scale of the graphs may vary dramatically, we first normalize all base graph adjacency matrices as transition matrices and learn a consensus transition matrix from them. Since the transition matrix has a clear probabilistic interpretation, we use the Kullback-Leibler divergence for consensus measuring. When selecting the features, we impose a weight on each feature and try to transform the weighted data into a subspace which can well preserve the intrinsic structure represented by the consensus graph. This procedure repeats until convergence.

To integrate the multiple graph learning and feature selection into a unified framework as introduced before, we carefully design a non-convex objective function. To optimize it, we propose a block coordinate descent algorithm and prove its convergence. We conduct extensive experiments on benchmark data sets by comparing our algorithm with several state-of-the-art unsupervised feature selection methods, and the experimental results show that ours outperforms these state-of-the-art methods.

The paper is organized as follows. Section 2 describes some related work. Section 3 presents in detail the main algorithm of our method. Section 4 shows the experimental results, and Section 5 concludes the paper.

2. Related work

To handle high dimensional data, many feature learning methods are proposed. One kind of feature learning method is feature extraction [25,26]. Feature extraction learns a projection to map the data from high dimensional feature space to a low dimensional space. For example, in [27], a multilinear principal component analysis was proposed to project the original image data to a low dimensional space; a sparse discriminant projection method was provided in [28]; most recently, Lai et al. [29] proposed a joint learning framework which could simultaneously extract features and learn the subspace. Although feature extraction has been demonstrated promising performance, feature selection, which is another kind of feature learning method, has better interpretability because it keeps the semantic meaning of the features [11]. Moreover, the cost of feature collection for learning can be reduced by feature selection because that we only need to collect the selected features in feature selection method rather than use all the features for projection as feature extraction does [11].

In feature selection, unsupervised feature selection is a more challenging problem due to the absence of labels, and thus has attracted considerable attention. Unsupervised feature selection methods try to select features which can well preserve the intrinsic structure of data. For example, Zhu et al. [30] selected features which can reconstruct the original data well; Wang et al. [31] proposed unsupervised feature selection method to preserve the pseudo-labels generated by matrix factorization; Zhou et al. [32] selected the features to preserve the balance structure of data. Besides these structures, another kind of popular representation of such intrinsic structure is graph. Therefore, many methods construct the graph from data and select features to preserve the graph structure. As introduced before, the graph based feature selection methods can be roughly categorized into two classes: (1) using a pre-defined graph; (2) learning an adaptive graph.

In the first class, the feature selection methods often construct a pre-defined graph such as heat kernel graph and cosine graph, and then select features which can preserve such graph structure well. For example, Zhao et al. [33] constructed heat kernel graph for feature selection; Yang et al. [19] applied local total scatter and between-class scatter matrix to select features; Zhao et al. [20] used the pairwise similarity graph for feature selection; Zhu et al. [21] proposed a co-regularized method using the heat kernel to construct the similarity matrix; Du et al. [34] constructed a k-nn graph for each feature and ranked the features by the linear reconstruction weights.

In the second class, the methods construct an adaptive graph in the procedure of feature selection, i.e., the structure of graph changes with selected features. For example, Du et al. [35] learned an adaptive graph for feature selection by preserving the global and local structure; Nie et al. [17] adaptively learned the local structure from the results of feature selection; Zhang et al. [36] constructed a hypergraph from all features to characterize the high-order similarities of data and selected features by the hypergraph clustering, and then they further proposed an adaptive hypergraph learning method to jointly learn the hypergraph and select features [37]. Fan et al. [18] proposed an unsupervised discriminant feature selection method which constructed the graph with pseudo-labels obtained by the results of subspace clustering; Zhu et al. [38] applied subspace clustering to learn the similarity matrix to guide the feature selection; Luo et al. [5] constructed the adaptive graph with structure regularization; Zheng et al. [22] learned a low rank structure for feature selection; Li et al. [39] proposed a generalized uncorrelated regression with adaptive graph for feature selection.

Both the classes of methods only use a single graph for feature selection, which may ignore the abundant information in the data. Therefore, we propose a feature selection method with adaptive multiple graph learning. Note that, our method is significantly different from the existing multiple graph feature selection method [40]. Firstly, that is a two-step method, i.e., it first linearly combines multiple graphs to obtain a new Laplacian matrix and then applies it to select features. Therefore, the two tasks (graph combining and feature selection) cannot be boosted by each other like our method. Secondly, in that method, the weight of each graph is set manually as hyper-parameters, which makes it difficult to handle data which contains more than two graphs. In our method, the weights and the consensus graph are both learned automatically in the process of feature selection.

It is worthy to mention another related fields called multi-view feature selection. These methods [41–44] select features from multiple views by integrating the information in each view. For example, Wang et al. [41] proposed a multi-view feature selection to integrate the features in all views and learn the weight for each feature via a joint structured sparsity-inducing norm; Liu et al. [42] presented a k-means based robust multi-view feature selection method. Since in each view we can construct a graph, some graph based multi-view feature selection methods are similar to the multiple graph feature selection. For example, Wang et al. [43] linearly combined all Laplacian matrices of multiple views and selected features with the consensus Laplacian matrix. Different from this method which linearly combines all graphs, our method learns a non-parametric consensus graph in such a way that we can effectively enlarge the region from which an optimal graph can be chosen for feature selection [45,46], i.e., we have a great chance to learn a better consensus graph.

3. Feature selection with multiple graphs

In this section, we introduce our feature selection method with adaptive multiple graph learning. Firstly, we introduce some notations in this paper. We use a bold uppercase character to denote a matrix and a bold lowercase character to denote a vector. For an arbitrary matrix $\mathbf{M} \in \mathbb{R}^{r\times s}$, $\mathbf{M}_{i\cdot}$ denotes its ith row, $\mathbf{M}_{\cdot i}$ denotes its ith column, and M_{ij} denotes its (i, j)th element.

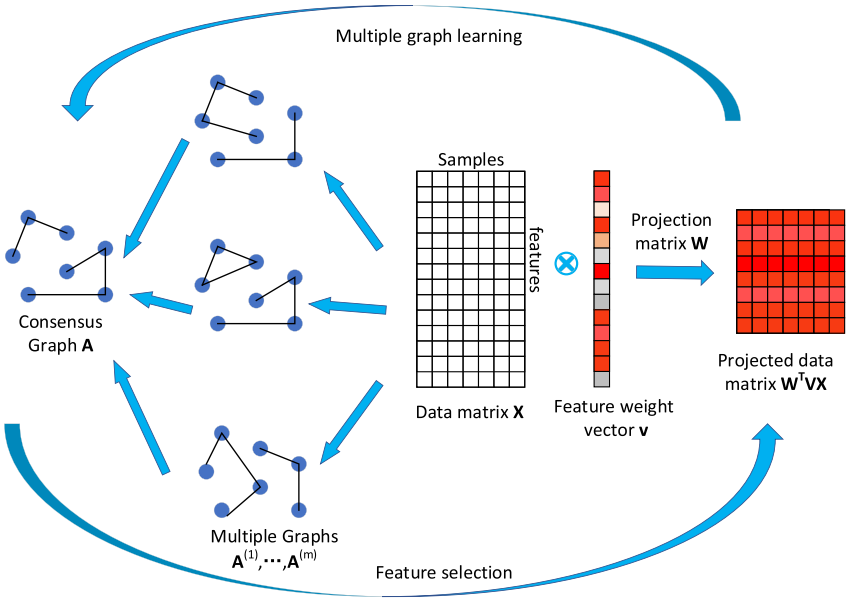


Fig. 1. The framework of our method. We use the data matrix $\mathbf{X}$ and the multiple base graphs $\mathbf{A}^{(1)}, \cdots, \mathbf{A}^{(m)}$ as input. Our framework is an iteration schema. We first use weight vector $\mathbf{v}$ to select features with the consensus graph $\mathbf{A}$ and then apply the selected features to learn the consensus graph $\mathbf{A}$ with the base graphs $\mathbf{A}^{(1)}, \cdots, \mathbf{A}^{(m)}$. This process repeats until convergence.

3.1. Framework

In our method, we seamlessly integrate the multiple graph learning and feature selection into a unified framework, for the reason that, we hope the two tasks can be boosted by each other. On the one hand, a clearer intrinsic graph structure can guide us to select more informative features; and on the other hand, the more informative features can be helpful to construct a better graph structure. In more details, we construct multiple graphs from data and learn a consensus graph from these multiple graphs; meanwhile, we use a feature weight vector to impose the weight on each feature and project the weighted data matrix into a low dimensional space to preserve the structure of the consensus graph. We jointly learn the consensus graph and select features until it converges. Fig. 1 shows the framework of our method.

3.2. Multiple graph learning

Let $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$ denotes a data set matrix containing n instances with d features, whose columns correspond to instances and rows to features. The task of feature selection is to select the most informative features from the original d features. In unsupervised feature selection, since the labels are absent, we should find the features which can preserve the intrinsic structure well. To this end, we make use of multiple graph adjacency matrices of data $\mathbf{A}^{(1)}, \cdots, \mathbf{A}^{(m)} \in \mathbb{R}^{n \times n}$, where $A_{ij}^{(k)} \geq 0$ denotes the (i, j)th element in $\mathbf{A}^{(k)}$. $A_{ij}^{(k)} > 0$ means the ith instance and the jth instance are connected by an edge in the kth graph and $A_{ij}^{(k)} = 0$ otherwise. Note that, in our method we use the graphs which do not contain the self-connections, i.e., $A_{ii}^{(k)} = 0$. These graphs can be generated in standard ways. For example, we can use Euclidean distance or cosine similarity to generate distance/similarity matrix and obtain k-nn graph directly from the distance/similarity matrix, or we can directly use the input multiple graphs when we handle multiple graph data. In our framework, we learn the intrinsic structure from these m graph matrices and adopt it to find the informative features.

Since the scale of the graphs may vary dramatically, we need to normalize all graphs. In our method, we normalize all input graph adjacency matrices by $\mathbf{A}^{(k)} \leftarrow \left(\mathbf{D}^{(k)}\right)^{-1}\mathbf{A}^{(k)}$ as transition matrices, where $\mathbf{D}^{(k)}$ is a diagonal matrix whose ith diagonal element $D_{ii}^{(k)} = \sum_j A_{ij}^{(k)}$. After the normalization, we have $\sum_j A_{ij}^{(k)} = 1$. It is easy to verify that $\mathbf{A}^{(k)}$ is a transition matrix, i.e., $A_{ij}^{(k)}$ indicates the probability of jumping in one step from the ith instance to the jth instance in a Markov random walk in the kth graph. Obviously, the larger $A_{ij}^{(k)}$ is, the more probably the ith instance jumps to the jth instance, the more probably the ith instance and the jth instance has a connection.

Since we aim to learn a consensus transition matrix $\mathbf{A}$ from $\mathbf{A}^{(1)}, \cdots, \mathbf{A}^{(m)}$, we should minimize the disagreement among these matrices. Considering that the elements in each row (i.e. $\mathbf{A}_{i.}^{(k)}$) have a clear probabilistic interpretation, we use the Kullback-Leibler divergence for consensus measuring. More formally, we minimize $\sum_k \alpha_k^2 \sum_i \mathrm{KL}(\mathbf{A}_{i.}^{(k)}, \mathbf{A}_{i.})$, where α_k is the weight of the kth graph. Taking the definition of Kullback-Leibler divergence into it, we minimize the following objective function:

$$\begin{aligned} \min_{\boldsymbol{\alpha}, \mathbf{A}} \quad & \sum_{k=1}^{m} \alpha_k^2 \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij}^{(k)} \log \frac{A_{ij}^{(k)}}{A_{ij}} \\ s.t. \quad & \sum_{j=1}^{n} A_{ij} = 1, \quad 0 \leq A_{ij} \leq 1, \quad A_{ii} = 0 \\ & \sum_{k=1}^{m} \alpha_k = 1, \quad \alpha_k \geq 0. \end{aligned} \tag{1}$$

where the first constraint makes sure that $\mathbf{A}_{i.}$ is a probability distribution and $A_{ii} = 0$ makes the vertices not self-connection; and the constraint $\sum_{k=1}^{m} \alpha_k = 1$ is to make sure that all weights sum up to 1. Note that the Kullback-Leibler divergence is asymmetry and we minimize $\mathrm{KL}(\mathbf{A}_{i.}^{(k)}, \mathbf{A}_{i.})$ instead of $\mathrm{KL}(\mathbf{A}_{i.}, \mathbf{A}_{i.}^{(k)})$. If we minimize $\mathrm{KL}(\mathbf{A}_{i.}, \mathbf{A}_{i.}^{(k)})$, A_{ij} must be zero as long as one of $A_{ij}^{(1)}, \cdots, A_{ij}^{(m)}$ is zero because $A_{ij}^{(k)}$ appears in denominator. Obviously, it is not what we really want.

3.3. Feature selection

To select the features, we define a d-dimension vector $\mathbf{v} = [v_1, v_2, \cdots, v_d]$ to indicate the weights of features, i.e., the larger v_i is, the more important the ith feature is. Then we define a diagonal weight matrix $\mathbf{V}$ whose ith diagonal element $V_{ii} = \sqrt{v_i}$. Here, the square root is for convenience of computation. To impose the weight on the instances, we can get the weighted data matrix as $\mathbf{VX}$.

Since $\mathbf{A}$ is learned from m graphs and indicates the intrinsic structure of the data, we wish the selected features can preserve such structure. Here we use a transformation matrix $\mathbf{W} \in \mathbb{R}^{d\times c}$ to project all weighted instances $\mathbf{VX}$ from original space into a new space by solving the following problem:

$$\begin{aligned} \min_{\mathbf{W},\mathbf{V}} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} \left\| \mathbf{W}^T\mathbf{V}\mathbf{x}_i - \mathbf{W}^T\mathbf{V}\mathbf{x}_j \right\|_2^2 A_{ij} + \lambda_1 \|\mathbf{W}\|_F^2, \\ s.t. \quad & \mathbf{W}^T\mathbf{V}\mathbf{X}\mathbf{X}^T\mathbf{V}\mathbf{W} = \mathbf{I}, \\ & \sum_{i=1}^{d} v_i = 1, \quad v_i \geq 0. \end{aligned} \tag{2}$$

where λ_1 is a balancing parameter for the ℓ_2 regularized term. The first term is to preserve the intrinsic structure in the projected space. In more detail, for each weighted instance $\mathbf{V}\mathbf{x}_i$, one of the other weighted instance $\mathbf{V}\mathbf{x}_j$ is considered as the neighborhood of $\mathbf{V}\mathbf{x}_i$ with probability A_{ij}. It can be found that by minimizing Eq. (2), a large A_{ij} will lead to a small $\|\mathbf{W}^T\mathbf{V}\mathbf{x}_i - \mathbf{W}^T\mathbf{V}\mathbf{x}_j\|_2^2$ which means the embedded values of $\mathbf{V}\mathbf{x}_i$ and $\mathbf{V}\mathbf{x}_j$ (i.e. $\mathbf{W}^T\mathbf{V}\mathbf{x}_i$ and $\mathbf{W}^T\mathbf{V}\mathbf{x}_j$) should be close. The first constraint can remove the redundant features and avoid the trivial solution, and the second constraint makes the weights in the range [0,1].

3.4. Final objective function

To integrate the multiple graph learning and feature selection into a unified framework, we combine Eqs. (1) and (2) to obtain the following objective function:

$$\begin{aligned} \min_{\mathbf{W},\mathbf{A},\mathbf{V},\boldsymbol{\alpha}} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} \left\| \mathbf{W}^T\mathbf{V}\mathbf{x}_i - \mathbf{W}^T\mathbf{V}\mathbf{x}_j \right\|_2^2 A_{ij} + \lambda_1 \|\mathbf{W}\|_F^2 \\ & + \lambda_2 \sum_{k=1}^{m} \alpha_k^2 \sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}^{(k)} \log \frac{A_{ij}^{(k)}}{A_{ij}}, \\ s.t. \quad & \mathbf{W}^T\mathbf{V}\mathbf{X}\mathbf{X}^T\mathbf{V}\mathbf{W} = \mathbf{I}, \\ & \sum_{i=1}^{d} v_i = 1, \quad v_i \geq 0, \\ & \sum_{k=1}^{m} \alpha_k = 1, \quad \alpha_k \geq 0, \\ & \sum_{j=1}^{n} A_{ij} = 1, \quad 0 \leq A_{ij} \leq 1, \quad A_{ii} = 0. \end{aligned} \tag{3}$$

where λ_2 is another balancing parameter.

In this framework, the learned graph matrix $\mathbf{A}$ is a non-parametric consensus graph instead of explicitly combining graphs and optimizing structured (e.g. linear) compositions of the graph matrices. Therefore, the learned graph can be as flexible as possible to fit the complex data. Moreover, multiple graph learning and feature selection can be iteratively boosted by each other.

3.5. Optimization

Firstly, since the variable $\mathbf{W}$ and $\mathbf{V}$ are entangled, we use a variable transformation which replaces $\mathbf{VW}$ by $\boldsymbol{\Phi}$ to simplify the optimization. By using this transformation, we rewrite the objective function as:

$$\begin{aligned} \min_{\boldsymbol{\Phi},\mathbf{A},\mathbf{v},\boldsymbol{\alpha}} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} \left\| \boldsymbol{\Phi}^T\mathbf{x}_i - \boldsymbol{\Phi}^T\mathbf{x}_j \right\|_2^2 A_{ij} + \lambda_1 \mathrm{tr}(\boldsymbol{\Phi}^T diag(\mathbf{v})^{-1}\boldsymbol{\Phi}) \\ & + \lambda_2 \sum_{k=1}^{m} \alpha_k^2 \sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}^{(k)} \log \frac{A_{ij}^{(k)}}{A_{ij}}, \\ s.t. \quad & \boldsymbol{\Phi}^T\mathbf{X}\mathbf{X}^T\boldsymbol{\Phi} = \mathbf{I}, \\ & \sum_{i=1}^{d} v_i = 1, \quad v_i \geq 0, \\ & \sum_{k=1}^{m} \alpha_k = 1, \quad \alpha_k \geq 0, \\ & \sum_{j=1}^{n} A_{ij} = 1, \quad 0 \leq A_{ij} \leq 1, \quad A_{ii} = 0. \end{aligned} \tag{4}$$

where $diag(\mathbf{v})$ means a diagonal matrix whose diagonal vector is $\mathbf{v}$.

Since Eq. (4) involves four groups of variables, we optimize it in a block coordinate descent schema. In more detail, we optimize one group of variables while fixing the other variables. This procedure repeats until convergence.

3.5.1. Optimizing $\boldsymbol{\Phi}$ by fixing $\mathbf{v}$, $\mathbf{A}$ and $\boldsymbol{\alpha}$

When other variables are fixed, Eq. (4) is rewritten as follows:

$$\begin{aligned} \min_{\boldsymbol{\Phi}} \quad & \mathrm{tr}(\boldsymbol{\Phi}^T\mathbf{X}\mathbf{L}\mathbf{X}^T\boldsymbol{\Phi}^T) + \lambda_1 \mathrm{tr}(\boldsymbol{\Phi}^T diag(\mathbf{v})^{-1}\boldsymbol{\Phi}), \\ s.t. \quad & \boldsymbol{\Phi}^T\mathbf{X}\mathbf{X}^T\boldsymbol{\Phi} = \mathbf{I}. \end{aligned} \tag{5}$$

where $\mathbf{L} = \mathbf{D} - (\mathbf{A} + \mathbf{A}^T)/2$ and $\mathbf{D}$ is a diagonal matrix whose diagonal element $D_{ii} = \sum_j (A_{ij} + A_{ji})/2$.

When optimizing $\boldsymbol{\Phi}$, Eq. (5) can be solved by generalized eigenvalue decomposition. However, the time complexity of generalized eigenvalue decomposition is $O(d^3 + nd^2)$ and is very time consuming because in feature selection tasks the number of features d is often large.

To address this problem, we use a two-step optimization inspired from Du and Co-authors [35,47]. Define a matrix $\mathbf{Y} \in \mathbb{R}^{n\times c}$ whose columns are eigenvectors of $\mathbf{L}$, i.e., $\mathbf{L}\mathbf{Y}_{,i} = \gamma_i \mathbf{Y}_{,i}$ where γ_i is one of the eigenvalues of $\mathbf{L}$. If we can find $\boldsymbol{\Phi}' \in \mathbb{R}^{d\times c}$ such that $\mathbf{X}^T\boldsymbol{\Phi}' = \mathbf{Y}$, then each column of $\boldsymbol{\Phi}'$ is an eigenvector of the generalized eigenvalue decomposition problem according to Du and Co-authors [35,47].

Therefore, we can solve $\boldsymbol{\Phi}$ in Eq. (5) by the two-step optimization: firstly, we solve the eigenvalue decomposition problem $\mathbf{L}\mathbf{Y} = \boldsymbol{\Gamma}\mathbf{Y}$ to get $\mathbf{Y}$; and secondly, we find $\boldsymbol{\Phi}$ by optimizing the following problem:

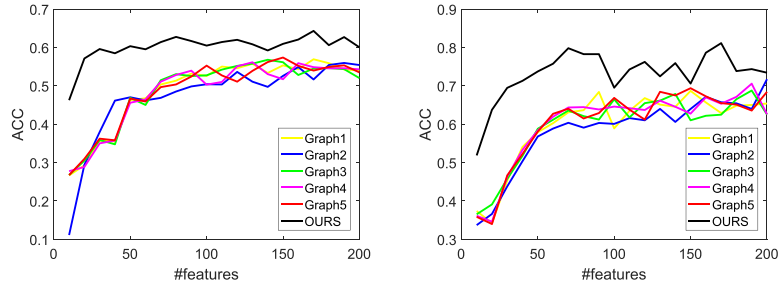
$$\min_{\boldsymbol{\Phi}} \left\| \mathbf{Y} - \mathbf{X}^T\boldsymbol{\Phi} \right\|_F^2 + \lambda_1 \mathrm{tr}(\boldsymbol{\Phi}^T diag(\mathbf{v})^{-1}\boldsymbol{\Phi}). \tag{6}$$

Set the derivative of Eq. (6) w.r.t. $\boldsymbol{\Phi}$ to zero, we can get its closed form solution:

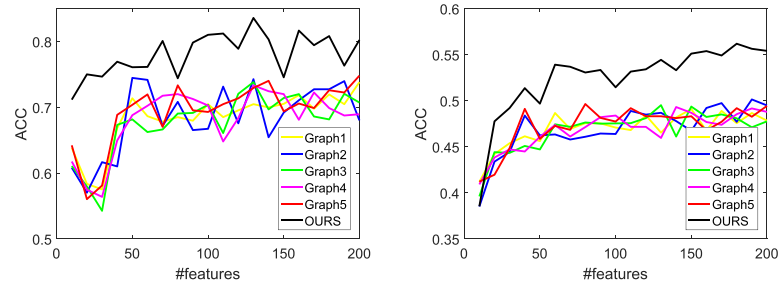
$$\boldsymbol{\Phi} = (\mathbf{X}\mathbf{X}^T + \lambda_1 diag(\mathbf{v})^{-1})^{-1}\mathbf{X}\mathbf{Y} \tag{7}$$

Note that $(\mathbf{X}\mathbf{X}^T + \lambda_1 diag(\mathbf{v})^{-1})$ is a d-by-d matrix and computing its inverse costs $O(d^3)$ time. For the data whose $d \gg n$, we can transform it by the Woodbury matrix identity as follows:

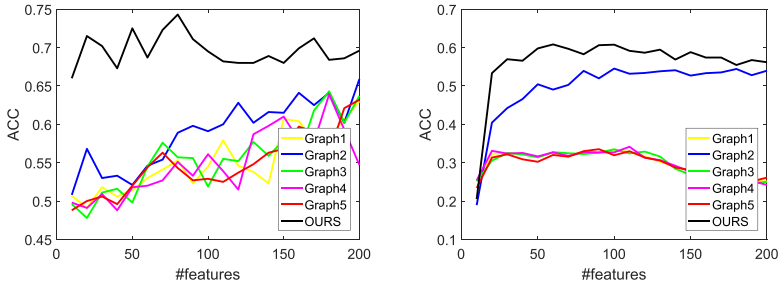
$$\begin{aligned} \boldsymbol{\Phi} &= (\mathbf{X}\mathbf{X}^T + \lambda_1 diag(\mathbf{v})^{-1})^{-1}\mathbf{X}\mathbf{Y} \\ &= \lambda_1^{-1}\Big(diag(\mathbf{v}) - \lambda_1^{-1} diag(\mathbf{v})\mathbf{X}\big(\mathbf{I} + \lambda_1^{-1}\mathbf{X}^T diag(\mathbf{v})\mathbf{X}\big)^{-1}\mathbf{X}^T diag(\mathbf{v})\Big)\mathbf{X}\mathbf{Y}. \end{aligned} \tag{8}$$



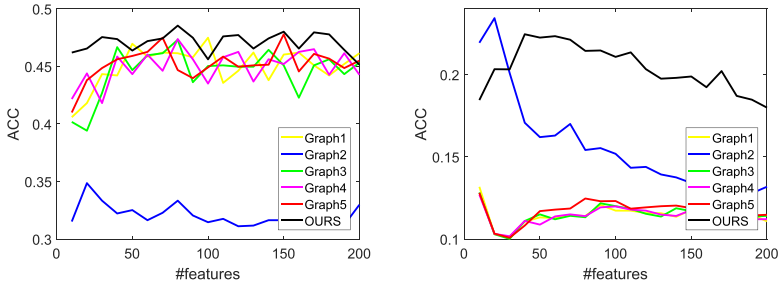
(a) Coil20 (b) Jaffe



(c) Lung (d) ORL



(e) Orlraws10P (f) PIE



(g) TOX-171 (h) YaleB

Fig. 2. ACC results compared with SingleGraph.

Obviously, the time complexity is reduced to $O(n^2d + ndc + n^3)$.

According to [35], optimizing $\mathbf{\Phi}$ by the two-step method can monotonically decrease the objective function.

3.5.2. Optimizing $\mathbf{v}$ by fixing $\mathbf{\Phi}$, $\boldsymbol{A}$ and $\boldsymbol{\alpha}$

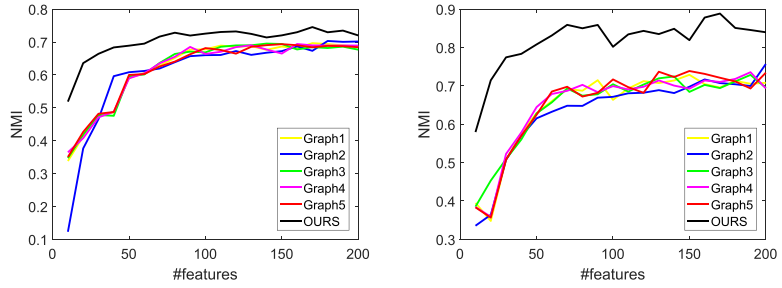
When optimizing $\mathbf{v}$, we need to minimize the following formula:

$$\min_{\mathbf{v}} \quad \mathrm{tr}(\mathbf{\Phi}^T diag(\mathbf{v})^{-1}\mathbf{\Phi}), \tag{9}$$

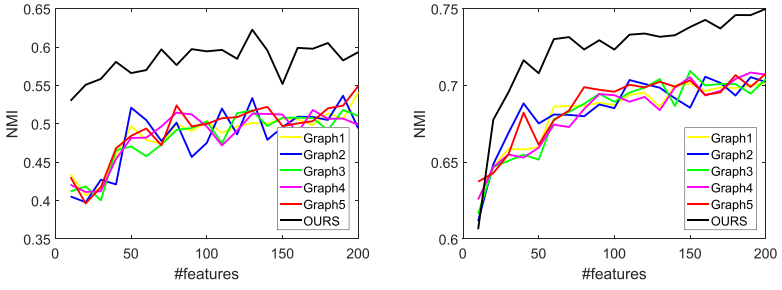
$$s.t. \quad \sum_{i=1}^{d} v_i = 1, \quad v_i \geq 0.$$

According to Cauchy-Schwarz Inequality, we have

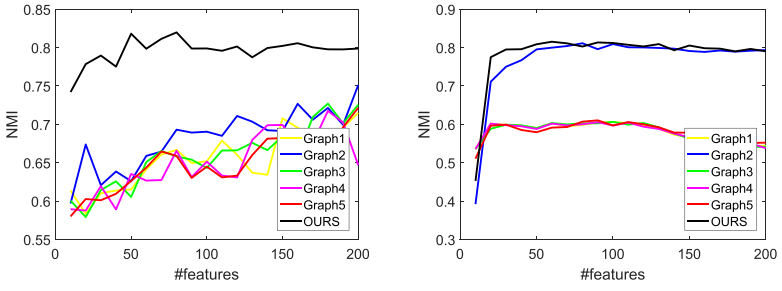
$$\begin{aligned} \mathrm{tr}(\mathbf{\Phi}^T diag(\mathbf{v})^{-1}\mathbf{\Phi}) &= \sum_i \frac{\sum_j \Phi_{ij}^2}{v_i} \\ &= \sum_i \frac{\sum_j \Phi_{ij}^2}{v_i} \cdot \sum_i v_i \\ &\geq \left(\sum_i \sqrt{\sum_j \Phi_{ij}^2} \right)^2. \end{aligned} \tag{10}$$



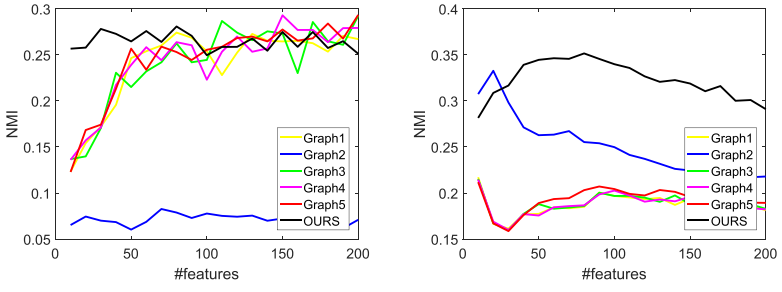
(a) Coil20 (b) Jaffe



(c) Lung (d) ORL



(e) Orlraws10P (f) PIE



(g) TOX-171 (h) YaleB

Fig. 3. NMI results compared with SingleGraph.

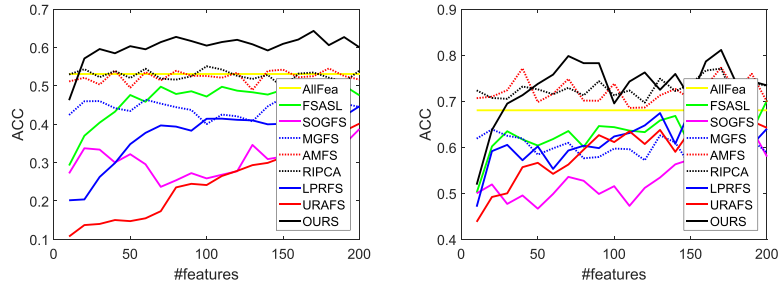
The equality in Eq. (10) holds when $v_i \propto \sqrt{\sum_j \Phi_{ij}^2}$. So the closed-form solution of Eq. (9) is

$$v_i = \frac{\sqrt{\sum_j \Phi_{ij}^2}}{\sum_i \sqrt{\sum_j \Phi_{ij}^2}}. \tag{11}$$

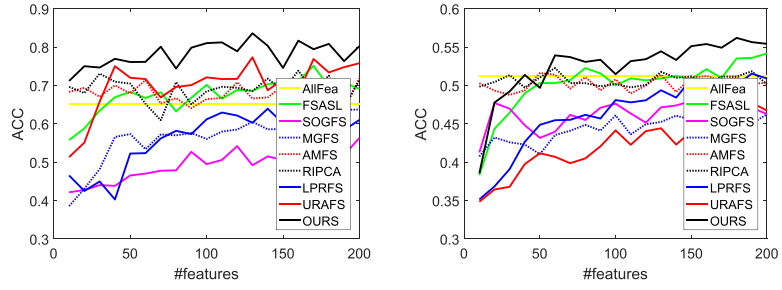
3.5.3. Optimizing $\boldsymbol{A}$ *by fixing* $\boldsymbol{\Phi}$, $\mathbf{v}$ *and* $\boldsymbol{\alpha}$

When $\boldsymbol{\Phi}$, $\mathbf{v}$ and $\boldsymbol{\alpha}$ are fixed, Eq. (3) can be rewritten as:

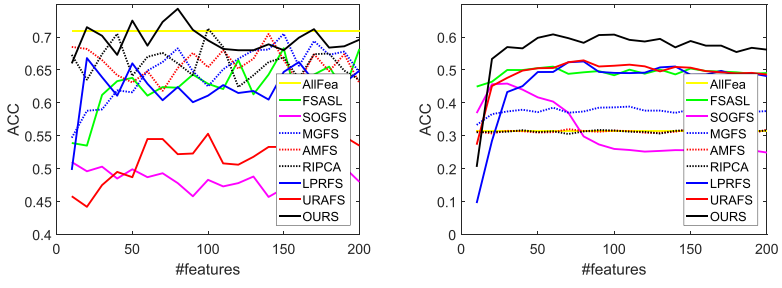
$$\begin{aligned} \min_{\mathbf{A}} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} B_{ij}A_{ij} - \lambda_2 \sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}\log(A_{ij}) \\ s.t. \quad & \sum_{j=1}^{n} A_{ij} = 1, \quad 0 \le A_{ij} \le 1, \quad A_{ii} = 0 \end{aligned} \tag{12}$$



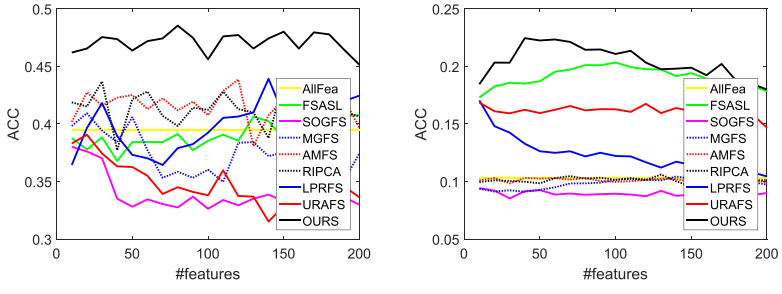
(a) Coil20 (b) Jaffe



(c) Lung (d) ORL



(e) Orlraws10P (f) PIE



(g) TOX-171 (h) YaleB

Fig. 4. ACC results compared with state-of-the-art feature selection methods.

where $B_{ij} = \left\| \mathbf{\Phi}^T \mathbf{x}_i - \mathbf{\Phi}^T \mathbf{x}_j \right\|_2^2$ and $C_{ij} = \sum_{k=1}^{m} \alpha_k^2 A_{ij}^{(k)}$. Obviously $C_{ii} = 0$ and $B_{ii} = 0$.

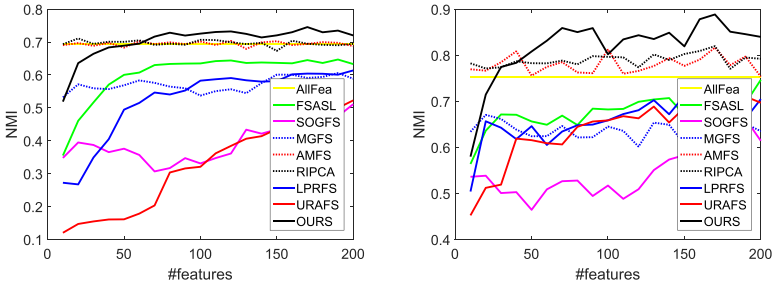
Since each row of Eq. (12) is decoupled, we focus on the ith row. Note that if $\mathbf{A}_{i\cdot}$ satisfies $\sum_{j=1}^{n} A_{ij} = 1$ and $A_{ij} \geq 0$, then $\mathbf{A}_{i\cdot}$ must also satisfy $A_{ij} \leq 1$. So we can drop the constraint $A_{ij} \leq 1$ safely. Moreover, since $B_{ii} = 0$, $C_{ii} = 0$ and the constraint makes $A_{ii} = 0$, we only need to consider A_{ij} for $i \neq j$. Introducing the Lagrange multipliers, we obtain the Lagrange function:

$$\mathcal{L} = \sum_{j \neq i} B_{ij} A_{ij} - \lambda_2 \sum_{j \neq i} C_{ij} \log(A_{ij}) + \theta \left(\sum_{j \neq i} A_{ij} - 1 \right) - \sum_{j \neq i} \mu_j A_{ij} \tag{13}$$

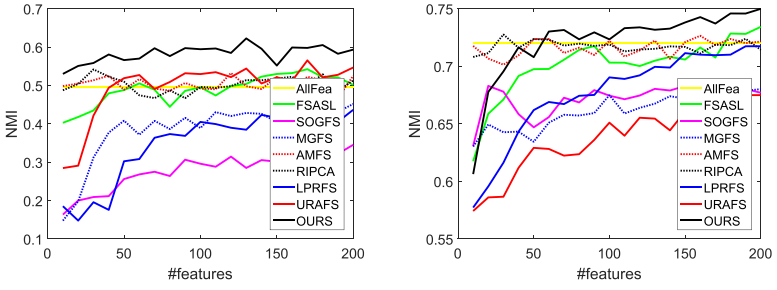
where θ and $\boldsymbol{\mu}$ are Lagrange multipliers.

Setting the partial derivative of $\mathcal{L}$ w.r.t. A_{ij} to zero, we get:

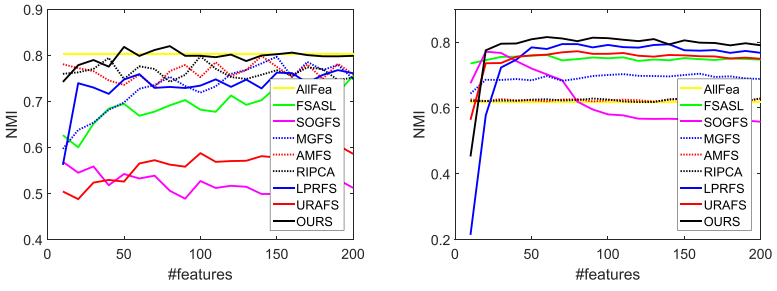
$$\frac{\partial \mathcal{L}}{\partial A_{ij}} = B_{ij} - \lambda_2 \frac{C_{ij}}{A_{ij}} + \theta - \mu_j = 0. \tag{14}$$



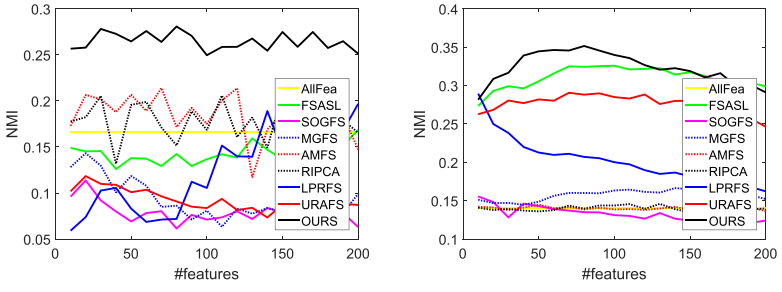
(a) Coil20 (b) Jaffe



(c) Lung (d) ORL



(e) Orlraws10P (f) PIE



(g) TOX-171 (h) YaleB

Fig. 5. NMI results compared with state-of-the-art feature selection methods.

Since Eq. (12) is convex and has a lower bound, a solution which satisfies the Karush-Kuhn-Tucker (KKT) conditions is the global optima of this subproblem. Considering its KKT conditions, we obtain:

$$\begin{cases} B_{ij} - \lambda_2 \frac{C_{ij}}{A_{ij}} + \theta - \mu_j = 0, \\ \sum_{j \neq i} A_{ij} = 1, \\ A_{ij} \geq 0, \\ \mu_j A_{ij} = 0, \\ \mu_j \geq 0. \end{cases} \tag{15}$$

For all j such that $C_{ij} \neq 0$, according to the KKT conditions, we get $A_{ij} = \frac{\lambda_2 C_{ij}}{B_{ij}+\theta-\mu_j}$ and $A_{ij} \neq 0$, so $\mu_j = 0$. Thus $A_{ij} = \frac{\lambda_2 C_{ij}}{B_{ij}+\theta}$. For all j such that $C_{ij} = 0$, we get $B_{ij} + \theta - \mu_j = 0$ and thus $\mu_j = B_{ij} + \theta \geq 0$.

Now we first find the smallest element in B_{ij} as B_{ip}, and then discuss in two cases:

1) In the case that $C_{ip} \neq 0$, we calculate θ by solving $\sum_{j:C_{ij}\neq 0} \frac{\lambda_2 C_{ij}}{B_{ij}+\theta} = 1$ in the range $(-B_{ip}, +\infty)$. We define a function $f(\theta) = \sum_{j:C_{ij}\neq 0} \frac{\lambda_2 C_{ij}}{B_{ij}+\theta}$ and have $\lim_{\theta \to -B_{ip}^+} f(\theta) \to +\infty$ and $\lim_{\theta \to +\infty} f(\theta) = 0$. Moreover, $f(\theta)$ is a monotone decreasing func-

tion in the range $(-B_{ip}, +\infty)$. So the equation $f(\theta) = 1$ has and only has one solution in the range $(-B_{ip}, +\infty)$. After obtaining the solution θ, we set $A_{ij} = \frac{\lambda_2 C_{ij}}{B_{ij}+\theta}$ and $\mu_j = 0$ for the j such that $C_{ij} \neq 0$. For all other j, we set $A_{ij} = 0$ and $\mu_j = B_{ij} + \theta$. It is easy to verify that $B_{ij} - \lambda_2 \frac{C_{ij}}{A_{ij}} + \theta - \mu_j = 0$, $\sum_{j \neq i} A_{ij} = 1$ and $\mu_j A_{ij} = 0$ in the KKT condition are satisfied. Since $\theta \geq -B_{ip}$ and B_{ip} is the smallest element in B_{ij}, for any $\mu_j \neq 0$, we have $\mu_j = B_{ij} + \theta \geq B_{ij} - B_{ip} \geq 0$. Similarly, for any $A_{ij} \neq 0$, we have $A_{ij} = \frac{\lambda_2 C_{ij}}{B_{ij}+\theta} \geq 0$. Therefore, all KKT conditions are satisfied.

2) In the case that $C_{ip} = 0$, we first calculate $f(-B_{ip})$. If $f(-B_{ip}) \geq 1$, we have the same result that $f(\theta) = 1$ has and only has one solution in the range $(-B_{ip}, +\infty)$. So we compute θ, A_{ij} and μ_j in the same way as in the first case and all KKT conditions are satisfied. If $f(-B_{ip}) < 1$, we set $\theta = -B_{ip}$ and for all j such that $C_{ij} = 0$, we set $\mu_j = B_{ij} + \theta = B_{ij} - B_{ip}$. For other j, we set $A_{ij} = \frac{\lambda_2 C_{ij}}{B_{ij} - B_{ip}}$ and $\mu_j = 0$. At last, $A_{ip} = 1 - \sum_{j:C_{ij} \neq 0} A_{ij} = 1 - \sum_{j:C_{ij} \neq 0} \frac{\lambda_2 C_{ij}}{B_{ij} - B_{ip}}$ and all other $A_{ij} = 0$. Since $f(-B_{ip}) < 1$, $A_{ip} = 1 - f(-B_{ip}) > 0$ and $\mu_{ip} = B_{ip} - B_{ip} = 0$. So all KKT conditions are also satisfied.

To sum up, A_{ij} calculated by the aforementioned method satisfies all the KKT conditions in Eq. (15) and thus such A_{ij} is the global optima of this subproblem.

3.5.4. Optimizing $\boldsymbol{\alpha}$ by fixing $\boldsymbol{\Phi}$, $\mathbf{v}$ and $\mathbf{A}$

When $\boldsymbol{\Phi}$, $\mathbf{v}$ and $\mathbf{A}$ are fixed, we rewrite Eq. (3) as follows:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \sum_{k=1}^{m} \alpha_k^2 c_k, \\ s.t. \quad & \sum_{k=1}^{m} \alpha_k = 1, \quad \alpha_k \geq 0. \end{aligned} \tag{16}$$

where $c_k = \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij}^{(k)} \log \frac{A_{ij}^{(k)}}{A_{ij}}$.

According to Cauchy-Schwarz Inequality, the global optima of Eq. (16) is

$$\alpha_k = \frac{c_k^{-1}}{\sum_i c_i^{-1}}. \tag{17}$$

According to Eq. (17), α_k is inversely proportional to c_k, i.e., the smaller c_k is, the larger α_k is. Note that, $c_k = \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij}^{(k)} \log \frac{A_{ij}^{(k)}}{A_{ij}}$ indicates the difference between the kth graph and the consensus graph, thus the small c_k, which lead to a large α_k, means the kth graph $\mathbf{A}^{(k)}$ is close to the consensus graph, i.e., the kth graph has a high quality. Therefore, in our method, the learned α_k can indeed represent the weight of the kth graph.

To sum up, we alternatively optimize $\boldsymbol{\Phi}$, $\mathbf{V}$, $\mathbf{A}$ and $\boldsymbol{\alpha}$ until it converges. Algorithm 1 summarizes the whole process. After obtaining $\mathbf{V}$, we select the l features corresponding to the largest l v_i's.

3.6. Convergence analysis

According to Du and Shen [35], updating $\boldsymbol{\Phi}$ makes the objective function decreasing monotonously. Computing $\mathbf{v}$, $\mathbf{A}$ and $\boldsymbol{\alpha}$ always find the global solution of each subproblem. So the objective function decreases in each iteration. Moreover, the objective function has a lower bound. Thus our method always converges. In fact, this algorithm converges very fast (within no more than five iterations in practice).

3.7. Time and space complexity

Since we need to save m $n \times n$ graph matrices $\mathbf{A}^{(1)}, \cdots, \mathbf{A}^{(k)}$ and the original data matrix $\mathbf{X}$, the space complexity is $O(mn^2 + nd)$. Furthermore, if the graph matrix is sparse, i.e., the average number of edges connected to a vertex is κ and $\kappa \ll n$, the space complexity can be reduced to $O(m\kappa n + nd)$.

In each iteration of Algorithm 1, we analyze the time complexity now. When optimizing $\boldsymbol{\Phi}$, as introduced before, the time complexity is $O(n^2 d + ndc + n^3)$. Optimizing $\mathbf{V}$ or computing Eq. (11) costs $O(cd)$ time. When optimizing $\mathbf{A}$, we first compute $\mathbf{B}$ in $O(cn^2 d)$ time and $\mathbf{C}$ in $O(n\kappa)$ time. Since we need to solve n subproblems, here we focus on the ith one. We need to solve a univariate equation $\sum_{j:C_{ij} \neq 0} \frac{\lambda_2 C_{ij}}{B_{ij}+\theta} = 1$. This equation has only one solution and we can solve it by standard root finding algorithm. We suppose that solving this equation costs $O(t)$ time. Then we need to compute A_i. Since we suppose that the average number of edges connected to a vertex is κ, i.e., there are κ non-zero values in the ith row of $\mathbf{A}$, we compute A_i in $O(t + \kappa)$ time. Therefore, we can compute $\mathbf{A}$ in $O(cn^2 d + (t + \kappa)n)$ time. Optimizing $\boldsymbol{\alpha}$ costs $O(nm\kappa)$ time since we need to compute m c_k's. To sum up, the time complexity is $O(cn^2 d + cnd + n^3 + m\kappa n + tn)$, thus the time complexity is linear with the number of features d. When handling the data set which $n \gg d$, we can compute $\boldsymbol{\Phi}$ directly by Eq. (7) instead of Eq. (8), thus time complexity is square with the number of instances n and cubic with d.

Algorithm 1 Feature selection with adaptive multiple graph learning.

Require: Data matrix $\mathbf{X}$, m graph adjacency matrices $\mathbf{A}^{(1)}, \cdots, \mathbf{A}^{(m)}$, parameters λ_1, λ_2.
Ensure: Feature weights $\mathbf{v}$.
1: Normalize $\mathbf{A}^{(k)}$ $(k = 1, \cdots, m)$ to make each row of it sums up to 1.
2: Initialize $\mathbf{A} = \sum_{i=1}^{m} \mathbf{A}^{(k)}/m$, and $\alpha_k = 1/m$.
3: **while** not converge **do**
4: Compute $\boldsymbol{\Phi}$ by Eq. (8).
5: Compute $\mathbf{v}$ by Eq. (11).
6: Compute $\mathbf{A}$ by solving Eq. (15).
7: Compute $\boldsymbol{\alpha}$ by Eq. (17).
8: **end while**

4. Experiments

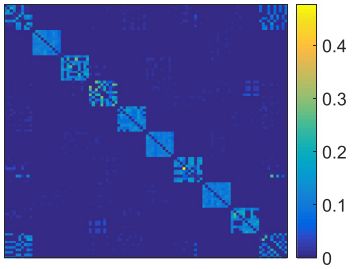
In this section, we compare our method with several state-of-the-art unsupervised feature selection methods on benchmark data sets.

4.1. Data sets

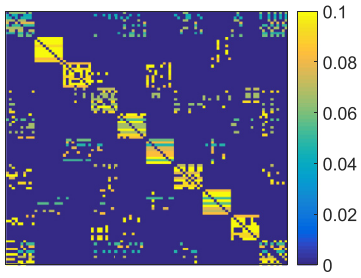
We collect 8 data sets, including Coil20[48], Jaffe [49], Lung[50], ORL[51], Orlraws10P[51], PIE[52], TOX-171[48], and YaleB[53]. The important statistics of these data sets are summarized in Table 1.

4.2. Experimental setup

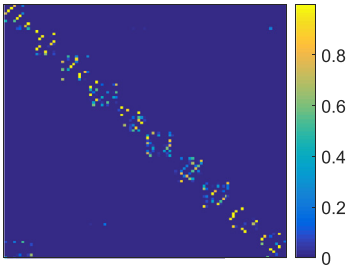
In our method, we construct 5 k-nn graphs with $k = 10$: one binary graph using Euclidean distance, i.e., the weight of an edge is fixed to 1; 3 heat kernel graphs with the edge weight $A_{ij} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2*t*d_0}}$ where d_0 is the average Euclidean distance of all instances and $t = \{0.1, 1, 10\}$; and one cosine graph with the edge weight $A_{ij} = cos(\mathbf{x}_i, \mathbf{x}_j)$. We use these graphs because all of them



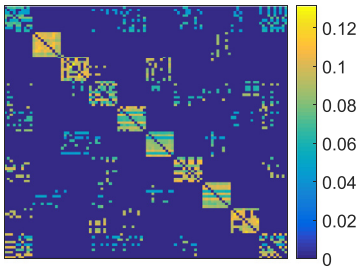
(a) Ours



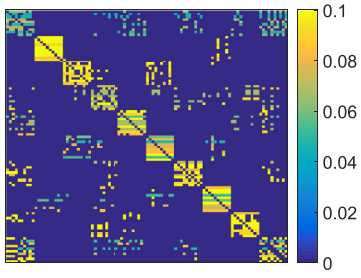
(b) Graph1



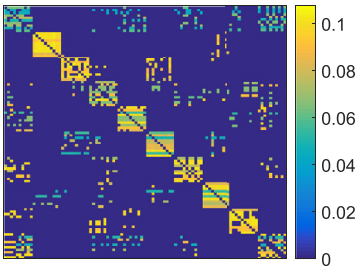
(c) Graph2



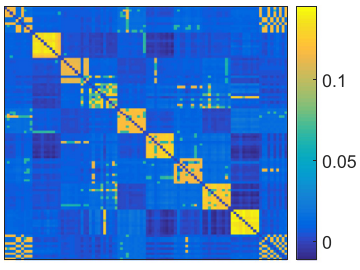
(d) Graph3



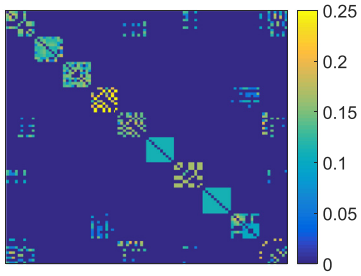
(e) Graph4



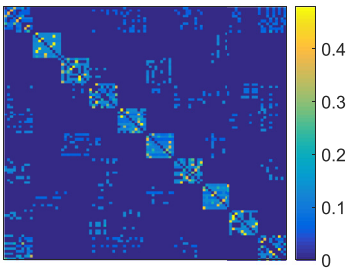
(f) Graph5



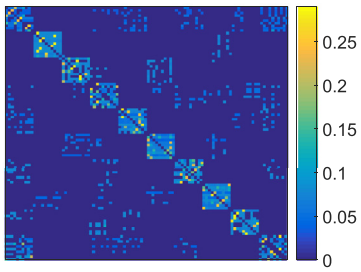
(g) FSASL



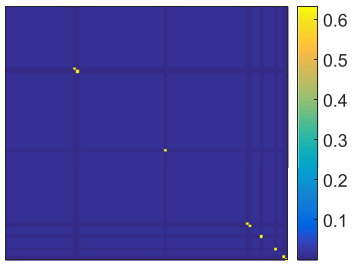
(h) SOGFS



(i) MGFS



(j) AMFS



(k) URAFS

Fig. 6. An illustration of the adjacency matrices of learned graphs in different graph based methods on Orlraws10P data set. (a) shows the learned consensus graph **A** in our method. (b)-(f) show the structure of the base graphs. (g)-(k) show the used or learned graph structure in the compared feature selection methods.

Table 1
Description of the data sets.

	#instances	#features	#classes
Coil20	1440	1024	20
Jaffe	213	676	10
Lung	203	3312	5
ORL	400	1024	40
Orlraws10P	100	10,340	10
PIE	1428	1024	68
TOX-171	171	5748	4
YaleB	2414	1024	38

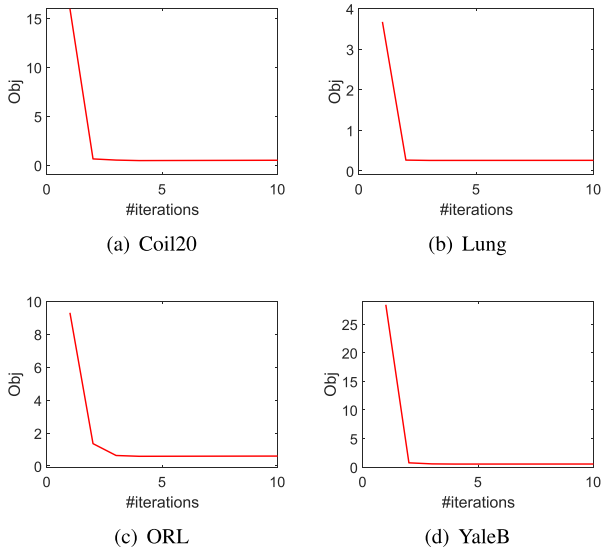


Fig. 7. Convergence curves of our method.

are very widely-used graphs in machine learning tasks. We compare our method with the following feature selection methods:

- **AllFea**. We use all features for clustering.
- **SingleGraph**. We use each of the five graphs introduced before as input in our method. Since it uses one single graph, the third term of Eq. (3) vanishes and Eq. (3) degenerates to the following form:

$$\begin{aligned} \min_{\mathbf{W},\mathbf{V}} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} \left\| \mathbf{W}^T\mathbf{V}\mathbf{x}_i - \mathbf{W}^T\mathbf{V}\mathbf{x}_j \right\|_2^2 A_{ij}^{(k)} + \lambda_1 \|\mathbf{W}\|_F^2 \\ s.t. \quad & \mathbf{W}^T\mathbf{V}\mathbf{X}\mathbf{X}^T\mathbf{V}\mathbf{W} = \mathbf{I}, \\ & \sum_{i=1}^{d} v_i = 1, \quad v_i \geq 0. \end{aligned} \tag{18}$$

  We solve this problem by the similar block coordinate descent method.
- **FSASL** [35]. It learns the adaptive global and local structure in the process of feature selection.
- **SOGFS** [17]. It learns a graph with optimal structure for unsupervised feature selection.
- **MGFS** [40]. It is a two-step multiple graph feature selection method, i.e., it first combines multiple graphs to obtain a consensus graph and then uses it to select features.
- **AMFS** [43]. It is a multi-view feature selection method and can be easily adapted to handle multiple graph setting by generating a graph for each view.
- **RIPCA** [54]. It is a rotational invariant projection method which uses $\ell_{2,1}$-norm to perform robust feature extraction.
- **LRPFS** [22]. It is a feature selection method which tries to preserve the low rank structure in the process of feature selection.
- **URAFS** [39]. This unsupervised feature selection method applies the generalized uncorrelated regression to learn an adaptive graph for feature selection.

With the selected features, we evaluate the performance in terms of k-means clustering by Accuracy (ACC) and Normalized Mutual Information (NMI). We run experiments 10 times and report the averaged results over different number of selected features (in the range {10, 20, ⋯, 200}). We tune the parameters of our method λ_1 and λ_2 in the range $[10^{-3}, 10^3]$ by the grid search. For other compared methods, we tune the parameters as suggested in their papers. For all methods on all data sets, the number of clusters is set to the true number of classes.

4.3. Experimental results

Firstly, we compare our method with SingleGraph method to evaluate whether it helps to use multiple graphs. Each SingleGraph method uses one of the 5 k-nn graphs as input and optimizes Eq. (18). The ACC and NMI results are shown in Figs. 2 and 3, respectively. In Figs. 2 and 3, Graph 1 represents the binary graph; Graphs 2–4 represent the heat kernel graph with $t = 0.1, 1, 10$ respectively; and Graph 5 represents the cosine graph. We can find that, on the most data sets, our method outperforms SingleGraph on all 5 graphs. It demonstrates that considering multiple graphs can indeed improve the performance of the method using only single graph.

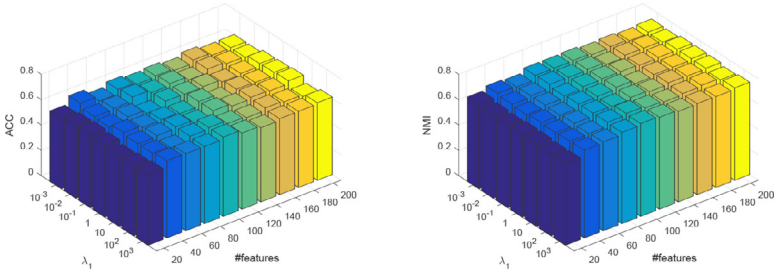
Then, we compare our method with the state-of-the-art unsupervised feature selection methods. Figs. 4 and 5 show the ACC and NMI results, respectively. The yellow horizontal line represents the result of AllFea which is the k-means result on all features. We can see that, on most data sets, our method can outperform the AllFea at most time. It demonstrates that our method can not only largely reduce the number of features used for clustering, but also often improve the clustering performance. It can also be found that our method outperforms the state-of-the-art feature selection methods on most data sets, which demonstrates the effectiveness of using adaptive multiple graph learning in feature selection. Even if compared with MGFS and AMFS which can also handle multiple graphs, our method still has a better performance. This reveals the advancement of our mechanism, i.e., learns an adaptive non-parametric consensus graph jointly with feature selection can improve the performance of feature selection.

To illustrate the effectiveness of our method, we show the adjacency matrices of the learned graphs of our method and other compared graph based methods on Orlraws10P data set in Fig. 6. Fig. 6(a) shows the values in the learned $\mathbf{A}$; Fig. 6(b)-(f) show the adjacency matrices of base graphs $\mathbf{A}^{(1)}, \cdots, \mathbf{A}^{(5)}$; Fig. 6(g)-(k) show the learned or used graphs in compared graph based feature selection methods. From Fig. 6, we can see that the structure of the learned graph in our method is clearer than not only all based graphs but also the graphs learned by the state-of-the-art methods. Thus, the learned graph can better uncover the cluster structure of data. This may be the main reason why our method can outperform both the single graph methods and the state-of-the-art feature selection methods.

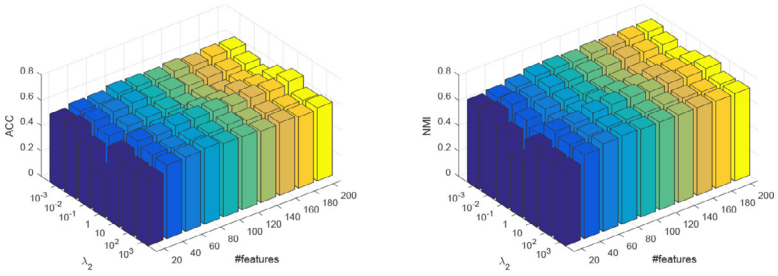
We show the algorithm convergence on Coil20, Lung, ORL and YaleB data sets in Fig. 7, and the results on other data sets are similar. The example results in Fig. 7 show that our method converges within a small number of iterations, which empirically demonstrates our claims in the previous section.

4.4. Parameter study

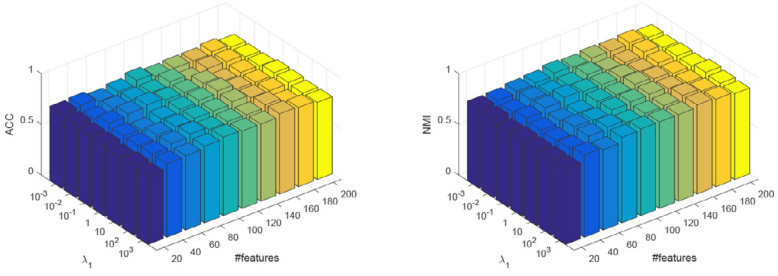
We explore the affect of the parameters on clustering performance by tuning parameters λ_1 and λ_2 in $[10^{-3}, 10^3]$. Fig. 8 shows



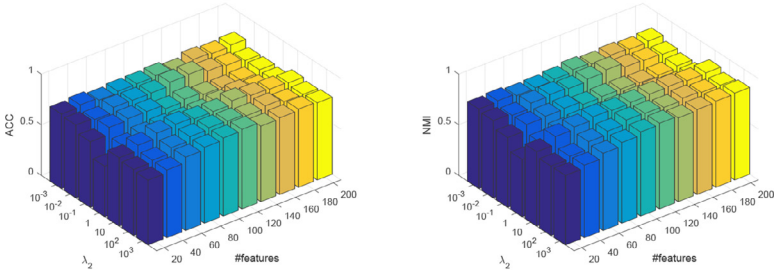
(a) ACC w.r.t. λ_1 on Coil20 (b) NMI w.r.t. λ_1 on Coil20



(c) ACC w.r.t. λ_2 on Coil20 (d) NMI w.r.t. λ_2 on Coil20



(e) ACC w.r.t. λ_1 on Jaffe (f) NMI w.r.t. λ_1 on Jaffe



(g) ACC w.r.t. λ_2 on Jaffe (h) NMI w.r.t. λ_2 on Jaffe

Fig. 8. ACC and NMI w.r.t λ_1, λ_2 on Coil20 and Jaffe data sets.

the results on Coil20 and Jaffe data sets and the results are similar on other data sets. The results show that the performance of our method is stable across a wide range of the parameters, thus we can choose the parameters easily.

5. Conclusion and future work

In this paper, we proposed a feature selection method with adaptive multiple graph learning. We made use of multiple graphs to learn an adaptive consensus graph to characterize the intrinsic structure of the data. To boost the structure learning and feature selection, we integrated them into a unified framework. We then presented a block coordinate descent method whose convergence is guaranteed to optimize the introduced objective function. Experimental results demonstrated that our method outperformed not only the ones using a single graph but also the state-of-the-art feature selection methods.

By integrating multiple graphs for feature selection, the proposed method can make full use of the complex structure of data. Moreover, it can also easily handle those data which are naturally performed in multiple graphs. However, there are some shortcomings of the proposed method, which we will try to address in the future. Firstly, although the time complexity is linear with d, it is still cubic with the number of instances. Therefore, it may be inappropriate to handle large scale data sets. In the future, we will consider this scalability issue and further reduce the time and space

complexity. Secondly, the proposed method uses all graphs for feature selection. Some graphs may not characterize the structure of data well. If we use these graphs, on the one hand, they may deteriorate the performance of feature selection; and on the other hand, they also bring more burdens for computation. To address this problem, we will consider the graph selection before feature selection, which can discard the bad graphs in advance.

Besides, it would be interesting to extend the proposed method to other feature learning tasks, such as feature extraction and dimension reduction. By considering multiple graphs of data, in these tasks, we can learn a more informative projection, which may better preserve the complex structure of data.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by the National Natural Science Fund of China grants 61806003, 61976129, and 61972001; the Key Natural Science Project of Anhui Provincial Education Department KJ2018A0010; and the National Natural Science Foundation of Shanxi grant 201801D221163.

References

[1] F. Nie, H. Huang, X. Cai, C.H. Ding, Efficient and robust feature selection via joint $\ell_{2,1}$-norms minimization, in: Advances in Neural Information Processing Systems, 2010, pp. 1813–1821.
[2] C. Hou, F. Nie, H. Tao, D. Yi, Multi-view unsupervised feature selection with adaptive similarity and view weight, IEEE TKDE 29 (9) (2017) 1998–2011.
[3] C. Tang, J. Chen, X. Liu, M. Li, P. Wang, M. Wang, P. Lu, Consensus learning guided multi-view unsupervised feature selection, Knowl. Based Syst. 160 (2018) 49–60.
[4] X. Zhu, S. Zhang, R. Hu, Y. Zhu, J. Song, Local and global structure preservation for robust unsupervised spectral feature selection, IEEE TKDE 30 (3) (2018) 517–529.
[5] M. Luo, F. Nie, X. Chang, Y. Yang, A.G. Hauptmann, Q. Zheng, Adaptive unsupervised feature selection with structure regularization, IEEE TNNLS 29 (4) (2018) 944–956.
[6] Y. Yang, J.O. Pedersen, A comparative study on feature selection in text categorization, in: ICML, 97, 1997, p. 35.
[7] K. Nigam, A.K. McCallum, S. Thrun, T. Mitchell, Text classification from labeled and unlabeled documents using em, Mach. Learn. 39 (2–3) (2000) 103–134.
[8] Y. Rui, T.S. Huang, S.-F. Chang, Image retrieval: current techniques, promising directions, and open issues, J. Vis. Commun. Image Represent. 10 (1) (1999) 39–62.
[9] Y. Saeys, I. Inza, P. Larrañaga, A review of feature selection techniques in bioinformatics, Bioinformatics 23 (19) (2007) 2507–2517.
[10] R. Tibshirani, Regression shrinkage and selection via the lasso, J. R. Stat. Soc. Series B (1996) 267–288.
[11] M. Fan, X. Chang, X. Zhang, D. Wang, L. Du, Top-k supervise feature selection via admm for integer programming, in: IJCAI, AAAI Press, 2017, pp. 1646–1653.
[12] Z. Xu, I. King, M.R.-T. Lyu, R. Jin, Discriminative semi-supervised feature selection via manifold regularization, IEEE Trans. Neural Netw. 21 (7) (2010) 1033–1047.
[13] J.G. Dy, C.E. Brodley, Feature selection for unsupervised learning, J. Mach. Learn. Res. 5 (Aug) (2004) 845–889.
[14] X. He, D. Cai, P. Niyogi, Laplacian score for feature selection, in: Advances in Neural Information Processing Systems, 2006, pp. 507–514.
[15] D. Cai, C. Zhang, X. He, Unsupervised feature selection for multi-cluster data, in: SIGKDD, ACM, 2010, pp. 333–342.
[16] X. Liu, L. Wang, J. Zhang, J. Yin, H. Liu, Global and local structure preservation for feature selection, IEEE TNNLS 25 (6) (2014) 1083–1095.
[17] F. Nie, W. Zhu, X. Li, et al., Unsupervised feature selection with structured graph optimization., in: AAAI, 2016, pp. 1302–1308.
[18] M. Fan, X. Chang, D. Tao, Structure regularized unsupervised discriminant feature analysis, in: AAAI, 2017, pp. 1870–1876.
[19] Y. Yang, H.T. Shen, Z. Ma, Z. Huang, X. Zhou, L21-norm regularized discriminative feature selection for unsupervised learning, IJCAI, 2011.
[20] Z. Zhao, L. Wang, H. Liu, J. Ye, On similarity preserving feature selection, IEEE TKDE 25 (3) (2013) 619–632.
[21] P. Zhu, Q. Xu, Q. Hu, C. Zhang, Co-regularized unsupervised feature selection, Neurocomputing 275 (2018) 2855–2863.
[22] W. Zheng, C. Xu, J. Yang, J. Gao, F. Zhu, Low-rank structure preserving for unsupervised feature selection, Neurocomputing 314 (2018) 360–370.
[23] F. Nie, J. Li, X. Li, et al., Parameter-free auto-weighted multiple graph learning: a framework for multiview clustering and semi-supervised classification., in: IJCAI, 2016, pp. 1881–1887.
[24] K. Zhan, F. Nie, J. Wang, Y. Yang, Multiview consensus graph clustering, IEEE Trans. Image Process. 28 (3) (2018) 1261–1270.
[25] P.N. Belhumeur, J.P. Hespanha, D.J. Kriegman, Eigenfaces vs. fisherfaces: recognition using class specific linear projection, IEEE Trans. Pattern Anal. Mach.Intell. (7) (1997) 711–720.
[26] Z. Lai, Y. Xu, J. Yang, J. Tang, D. Zhang, Sparse tensor discriminant analysis, IEEE Trans. Image Process. 22 (10) (2013) 3904–3915.
[27] Z. Lai, Y. Xu, Q. Chen, J. Yang, D. Zhang, Multilinear sparse principal component analysis, IEEE Trans. Neural Netw. Learn. Syst. 25 (10) (2014) 1942–1950.
[28] Z. Lai, Y. Xu, Z. Jin, D. Zhang, Human gait recognition via sparse discriminant projection learning, IEEE Trans. Circ. Syst. Video Technol. 24 (10) (2014) 1651–1662.
[29] Z. Lai, D. Mo, J. Wen, L. Shen, W.K. Wong, Generalized robust regression for jointly sparse subspace learning, IEEE Trans. Circ. Syst. Video Technol. 29 (3) (2018) 756–772.
[30] P. Zhu, W. Zuo, L. Zhang, Q. Hu, S.C. Shiu, Unsupervised feature selection by regularized self-representation, Pattern Recognit. 48 (2) (2015) 438–446.
[31] S. Wang, J. Tang, H. Liu, Embedded unsupervised feature selection, in: Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015.
[32] P. Zhou, J. Chen, M. Fan, L. Du, Y.-D. Shen, X. Li, Unsupervised feature selection for balanced clustering, Knowl. Based Syst. (2020) 105417.
[33] Z. Zhao, H. Liu, Spectral feature selection for supervised and unsupervised learning, in: Proceedings of the 24th international conference on Machine learning, ACM, 2007, pp. 1151–1157.
[34] L. Du, C. Ren, X. Lv, Y. Chen, P. Zhou, Z. Hu, Local graph reconstruction for parameter free unsupervised feature selection, IEEE Access 7 (2019) 102921–102930.
[35] L. Du, Y.-D. Shen, Unsupervised feature selection with adaptive structure learning, in: SIGKDD, ACM, 2015, pp. 209–218.
[36] Z. Zhang, E.R. Hancock, Hypergraph based information-theoretic feature selection, Pattern Recognit. Lett. 33 (15) (2012) 1991–1999.
[37] Z. Zhang, L. Bai, Y. Liang, E. Hancock, Joint hypergraph learning and sparse regression for feature selection, Pattern Recognit. 63 (2017) 291–309.
[38] P. Zhu, W. Zhu, Q. Hu, C. Zhang, W. Zuo, Subspace clustering guided unsupervised feature selection, Pattern Recognit. 66 (2017) 364–374.
[39] X. Li, H. Zhang, R. Zhang, Y. Liu, F. Nie, Generalized uncorrelated regression with adaptive graph for unsupervised feature selection, IEEE TNNLS (2018).
[40] X. Du, Y. Yan, P. Pan, G. Long, L. Zhao, Multiple graph unsupervised feature selection, Signal Process. 120 (2016) 754–760.
[41] H. Wang, F. Nie, H. Huang, Multi-view clustering and feature learning via structured sparsity, in: International Conference on Machine Learning, pp. 352–360.
[42] H. Liu, H. Mao, Y. Fu, Robust multi-view feature selection, in: International Conference on Data Mining, pp. 281–290.
[43] Z. Wang, Y. Feng, T. Qi, X. Yang, J.J. Zhang, Adaptive multi-view feature selection for human motion retrieval, Signal Process. 120 (2016) 691–701.
[44] R. Zhang, F. Nie, X. Li, X. Wei, Feature selection with multi-view data: a survey, Inf. Fusion 50 (2019) 158–167.
[45] P. Zhou, Y.-D. Shen, L. Du, Y. Fan, Incremental multi-view support vector machine, in: Proceedings of the 2019 SIAM International Conference on Data Mining, 2019, pp. 1–9.
[46] P. Zhou, Y.-D. Shen, L. Du, F. Ye, X. Li, Incremental multi-view spectral clustering, Knowl. Based Syst. 174 (2019) 73–86.
[47] D. Cai, X. He, J. Han, Spectral regression: a unified approach for sparse subspace learning, ICDM, 2007.
[48] J. Li, K. Cheng, S. Wang, F. Morstatter, T. Robert, J. Tang, H. Liu, Feature selection: a data perspective, arXiv:1601.07996 (2016).
[49] M.J. Lyons, J. Budynek, S. Akamatsu, Automatic classification of single facial images, IEEE Trans. Pattern Anal. Mach. Intell. 21 (12) (1999) 1357–1362.
[50] Z. Hong, J. Yang, Optimal discriminant plane for a small number of samples and design method of classifier on the plane, Pattern Recognit. 24 (4) (1991) 317–324.
[51] F. Samaria, A. Harter, Parameterisation of a stochastic model for human face identification, in: Workshop on Applications of Computer Vision, 1994, pp. 138–142.
[52] T. Sim, S. Baker, M. Bsat, The CMU pose, illumination, and expression (PIE) database, in: IEEE International Conference on Automatic Face and Gesture Recognition, 2002, pp. 53–58.
[53] K. Lee, J. Ho, D.J. Kriegman, Acquiring linear subspaces for face recognition under variable lighting, IEEE Trans. Pattern Anal. Mach. Intell. 27 (5) (2005) 684–698.
[54] Z. Lai, Y. Xu, J. Yang, L. Shen, D. Zhang, Rotational invariant dimensionality reduction algorithms, IEEE Trans. Cybern. 47 (11) (2017) 3733–3746.

Peng Zhou received the B.E. degree in computer science from University of Science and Technology of China in 2011 and Ph.D. degree in computer science from the Institute of Software, Chinese Academy of Sciences in 2017. He is currently a lecturer in Anhui University. His research interests include machine learning and data mining.

Liang Du received the B.E. degree in software engineering from Wuhan University in 2007, and Ph.D. degree in computer science from the Institute of Software, Chinese Academy of Sciences in 2013. He is currently a lecturer in ShanXi University. His research interests include machine learning, data mining and big data analysis.

Xuejun Li received the Ph.D. degree from Anhui University in 2008. He is currently a professor of School of Computer Science & Technology, Anhui University, China. His major research interests include intelligent software, cloud computing, and workflow systems.

Yi-Dong Shen is a professor of computer science in the State Key Laboratory of Computer Science at the Institute of Software, the Chinese Academy of Sciences. His main research interests include knowledge representation and reasoning, semantic web, and data mining.

Yuhua Qian received the M.S. and Ph.D. degrees from Shanxi University, Taiyuan, China, in 2005 and 2011, respectively. He is currently a Professor with the Key Laboratory of Computational Intelligence and Chinese Information Processing of Ministry of Education, Shanxi University. He has authored more than 70 papers in his research fields.