Joint Hypergraph Learning and Sparse Regression for Feature Selection

Zhihong Zhang[a], Lu Bai[b,*], Yuanheng Liang[a], Edwin Hancock[c]

[a]*Xiamen University, Xiamen, China*
[b]*Central University of Finance and Economics, Beijing, China.*
[c]*University of York, York, UK.*

Abstract

In this paper, we propose a unified framework for improved structure estimation and feature selection. Most existing graph-based feature selection methods utilise a static representation of the structure of the available data based on the Laplacian matrix of a simple graph. Here on the other hand, we perform data structure learning and feature selection simultaneously. To improve the estimation of the manifold representing the structure of the selected features, we use a higher order description of the neighbourhood structures present in the available data using hypergraph learning. This allows those features which participate in the most significant higher order relations to be selected, and the remainder discarded, through a sparsification process. We formulate a single objective function to capture and regularise the hypergraph weight estimation and feature selection processes. Finally, we present an optimization algorithm to recover the hyper graph weights and a sparse set of feature selection indicators. This process offers a number of advantages. First, by adjusting the hypergraph weights, we preserve high-order neighborhood relations reflected in the original data, which cannot be modeled by a simple graph. Moreover, our objective function captures the global discriminative structure of the features in the data. Comprehensive experiments on 9 benchmark data sets show that our method achieves statistically significant improvement over state-of-art feature selection methods, supporting the effectiveness of the proposed method.

*Corresponding author
Email address: bailucs@cufe.edu.cn (Lu Bai)

Keywords: Feature selection, Hypergraph learning, sparse regression

1. Introduction

Feature selection aims to locate an optimal set of features using a selection criterion. It is an important technique widely used in pattern analysis. It reduces data dimensionality by removing irrelevant and redundant features, and brings about a number of immediate benefits, such as speeding up a data mining algorithm, improving predictive accuracy, and enhancing comprehensibility. According to the way in which label information is utilized, feature selection algorithms can be categorized as a) supervised algorithms, b) unsupervised algorithms or c) semi-supervised algorithms. Examples of supervised feature selection algorithms include the Fisher Score (FScore) [1], similarity preserving feature selection (SPFS)[2], minimum redundancy maximum relevance (mRMR) [3], local-learning based feature selection (LLFS) [4], robust feature selection via $\ell_{2,1}$-norm minimization (L21RFS) [5] and the Trace ratio [6], which only use labeled training data for feature selection. When sufficient labeled training samples are to used, supervised feature selection is a reliable alternative, which selects discriminative features by exploiting class labels. However, labeling a large set of training samples manually is unrealistic in many real-world applications. In unsupervised feature selection on the other hand, there is no label information, and the features are selected which best preserve the data similarity or manifold structure. Examples include the Laplacian score (LapScore) [7], spectral feature selection (SPEC) [8], multi-cluster feature selection (MCFS) [9], joint embedding learning and sparse regression (JELSR) [10]. Recent work on semi-supervised learning has indicated that it is beneficial to leverage both labeled and unlabeled training data for data analysis. Motivated by the progress of semi-supervised learning, considerable effort has been devoted to semi-supervised feature selection. Recent reported algorithms include discriminative semi-supervised feature selection via manifold regularization (FS-Manifold) [11], locality sensitive semi-supervised feature selection (LSDF) [12], the spectral analysis of semi-supervised feature selection [13] and the noise insensitive trace ratio criterion (TRCFS)[14]. Usually, these methods use graph representations to characterize the

manifold structure.

However, there are two common problems with the aforementioned methods. First, the graph construction process is independent of a specific learning process. Once a graph is determined that characterizes the initial manifold structure of the data, it remains fixed in the following ranking or regression steps of feature selection. Therefore, the performance of feature selection is largely determined by the effectiveness of the graph construction. A typical example is the k-nearest neighbor graph used in Locality Preserving Projection (LPP) [15]. LPP first constructs a k-nearest neighbor graph (including its edge weights) based on the given raw data, and then seeks an optimal linear transformation with the aim to preserve such a neighborhood graph or the geometry of a given set of data. This initial graph is based on the characterization of "locality" which is unnecessary to be optimal, since it is difficult to set the parameters in advance (e.g., the neighborhood size and heat kernel width). In fact, these parameters have a significant impact on the ultimate performance of the algorithm. Second, in many situations the graph representation can lead to a substantial loss of information. This is because in real-world problems objects and their features tend to exhibit multiple relationships rather than simple pairwise ones. For example, consider the problem of classifying faces which are viewed under different lighting conditions. See Fig. 1 for an illustration. It is well known that images of the same objects may appear drastically different under different lighting conditions [16, 17]. In this scenario, the pairwise similarity measures for images of the same person may exhibit significant randomness. This misleading result is due to the fact that the set of images of a Lambertian surface under arbitrary lighting lies on a 3D subspace in the image space [18] where multiple relationships exist. As a result, higher order relations cannot be meaningfully characterized by pairwise similarity measures.

A natural way of remedying the information loss described above is to represent the data set as a hypergraph instead of a graph. Hypergraph representations allow vertices to be multiply connected by hyperedges and can hence capture multiple or higher order relationships between features. Due to their effectiveness in representing multiple relationships, hypergraph based methods have been applied to various practical problems, such as partitioning circuit netlists [19], clustering [20, 21], clustering cate-

Figure 1: Shown above are images of five persons under varying illumination conditions. Is it possible to group them into clusters based on pairwise similarity measure?

gorial data [22], and image segmentation [23]. For multi-label classification, Sun et al. [24] construct a hypergraph to exploit the correlation information contained in different labels. In this hypergraph, instances correspond to the vertices and each hyperedge includes all instances annotated with a common label. With this hypergraph representation, the higher-order relations among multiple instances sharing the same label can be explored. Following the theory of spectral graph embedding [25], they transform the data into a lower-dimensional space through a linear transformation, which preserves the instance-label relations captured by the hypergraph. The projection is guided by the label information encoded in the hypergraph and a linear Support Vector Machine (SVM) is used to handle the multi-label classification problem. Huang et al. [26] used a hypergraph cut algorithm [21] to solve the unsupervised image categorization problem, where a hypergraph is used to represent the complex relationships between unlabeled images based on shape and appearance features. Specifically, they first extract regions of interest (ROI) for each image, and then construct hyperedges among images based on shape and appearance features in their ROIs. Hyperedges are defined as either a) a group formed by each vertex (image) or b) its k-nearest neighbors (based on shape or appearance descriptors). The weight of each hyperedge is computed as the sum of the pairwise affinities within the hyperedge. In this way, the task of image categoriza-

tion is transferred into a hypergraph partition problem which can be solved using the hypergraph cut algorithm.

One common feature of these existing hypergraph representations is that they exploit domain specific and goal directed representations. Specifically, most of them are confined to uniform hypergraphs where each of the hyperedges have the same cardinality and therefore do not lend themselves to generalization. The reason for this lies in the difficulty in formulating a nonuniform hypergraph in a mathematically elegant way for the purpose of computation. There has yet to be a widely accepted and consistent way for representing and characterizing nonuniform hypergraphs, and this remains an open problem when exploiting hypergraphs for feature selection.

To address these shortcomings, an effective method for hypergraph construction is needed, such that the ambiguities of relational order can be overcome. In this paper, we improve the hypergraph construction approach presented above using a sparse representation model. Specifically, a hypergraph is constructed using each sample as a node, and a hyperedge includes a sample and its correlated samples, with the corresponding non-zero elements extracted in the sparse vector. Instead of generating a single hyperedge for each sample, we generate a group of hyperedges by varying regularization parameter values to give different sparsity solutions of the model. This makes our approach much more robust than previous hypergraph methods, because we do not need to tune the neighborhood size as a parameter. However, with this hypergraph construction approach, a large number of remaining hyperedges are generated with redundancy. In addition, they have different effects in classification accuracy. For example, hyperedges that are generated from samples close to the classification boundary may link samples from different classes. Since samples connected by a hyperedge are expected to be from the same class, the hyperedges that link samples from different classes will be less informative or may even have derogatory effects. Therefore, in order to modulate the effects of different hyperedges, we place a regularizer on the hyperedge weights. In this way, the effects of different hyperedges can be adaptively modulated and useless hyperedges can be discarded (i.e., the weights of redundant hyperedges will be 0), and thus, we can select the most effective hyperedges.

In this paper, we propose a unified learning framework which performs structure

learning and feature selection simultaneously. The structures are adaptively learned from the results of hypergraph learning, and the informative features are selected to preserve the refined structures of data. The hypergraph can well keep high-order neighborhood relationship reflected by the original data, which cannot be modeled by a simple graph. Moreover, rather than just targeting the locality preserving power characterized by hypergraph learning, our objective function also considers global discriminative structure of data. Concretely, global discriminative information in our framework is preserved by exploiting the underlying pairwise sample similarity. The sample similarity measure may introduce the discriminative information when the data labels are known. Comprehensive experiments on seven benchmark data sets show that our method achieves statistically significant improvement over state-of-art feature selection methods, suggesting the effectiveness of the proposed method.

2. Related Work

In this section, we first establish a list of the main notations used in the paper and summarized in Table.1. Then, we review some of the well-known algorithms for learning-based feature selection, all of which are closely related to our proposed method.

1) LapScore: Laplacian score [7] uses a k-nearest neighbor graph to model the local geometric structure of the data and selects the features most consistent with the graph structure. Consider a dataset $\mathbf{X} = [x_1, \ldots, x_n]^T$, in order to approximate the manifold structure of the dataset, a k-nearest neighbor graph is built, which contains an edge with weight w_g^{ij} between x_i and x_j if x_i is among the k nearest neighbors of x_j or conversely. There are different similarity based methods that can be used to determine the edge weights. In general, the Euclidean distance is widely used as similarity measure. Therefore, the element w_g^{ij} of the weight matrix $\mathbf{W_g}$ can be defined as below,

$$w_g^{ij} = \begin{cases} e^{-\frac{\|x_i - x_j\|^2}{t}}, & \text{if } x_i \text{ and } x_j \text{ are neighbors} \\ 0, & \text{otherwise.} \end{cases} . \quad (1)$$

where t is a suitable constant. A feature that is consistent with the graph structure can be thought of as the one for which two data points are close to each other if and only

Table 1: Important notations used in this paper and their definitions

d	The dimension of input data, i.e. the number of all features of input data.
n	The number of data points.
NoF	The number of selected features.
k	Dimensionality of embedding.
m	The number of hyperedges.
l	The number of selected labeled data out of all data $\mathbf{X}$.
$\mathbf{X}$	$\mathbf{X} = [x_1, \ldots, x_n]^T \in \Re^{n \times d}$ is the input data matrix. Each row $x_i \in \Re^d$ denotes a data point, for $i = 1, \ldots, n$.
$\mathbf{W_g}$	$\mathbf{W_g}$ is the weight matrix of graph where each edge weigh is represented by w_g^{ij}. Here we assume w_g^{ij} is symmetric where $w_g^{ij} = w_g^{ji}$
f_r	$f_r = (f_{r1}, \ldots, f_{rn})^T \in \Re^n$ is the r-th feature vector of data ($r = 1, \ldots, d$). It is also the r-th column of the data matrix $\mathbf{X}$, i.e., $\mathbf{X} = [f_1, \ldots, f_d]$.
$\mathbf{D}$	$\mathbf{D}$ is the diagonal degree matrix of graph where $\mathbf{D}_{ii} = \sum_j w_g^{ij}$
$\mathbf{Y}$	$\mathbf{Y} = [y_1, y_2, \ldots, y_n]^T \in \Re^{n \times k}$ is the data matrix of embedding
$\mathbf{W}$	$\mathbf{W} = [w_1, w_2, \ldots, w_k] \in \Re^{d \times k}$ is the transformation matrix
$\mathbf{D_e}$	The diagonal matrix of the hyperedge degrees
$\mathbf{D_v}$	The diagonal matrix of the hypergraph vertex degrees
$\mathbf{H}$	The incidence matrix of the hypergraph
$\mathbf{W_H}$	The diagonal weight matrix and its (i, i)-th element is the weight of the i-th hyperedge
$\hat{\mathbf{L}}_\mathbf{H}$	The normalized Laplacian matrix of hypergraph
$\mathbf{S}$	$\mathbf{S} \in \Re^{d \times k}$ is the sparse transformation matrix
$\mathbf{A}$	$\mathbf{A} \in \Re^{l \times n}$ is a binary selection matrix. It selects the labeled data out of all data $\mathbf{X}$
$\mathbf{K}$	$\mathbf{K}$ is a predefined similarity matrix.
$w(e)$	The weight of hyperedge e
$\delta(e)$	The degree of the hyperedge e

if there is an edge between these two points. Let f_{ri} denote the i-th sample of the r-th feature and $f_r = (f_{r1}, \ldots, f_{rn})^T$. To select a good feature, we need to minimize the following objective function:

$$SC_{Ls} = \frac{\sum_{ij}(f_{ri} - f_{rj})^2 w_g^{ij}}{Var(f_r)} \ . \tag{2}$$

where $Var(f_r)$ is the estimated variance of the r-th feature. Features with larger variance are preferred, as they are expected to have more representational power. Given $\mathbf{W_g}$, its corresponding degree matrix $\mathbf{D}_{ii} = \sum_j w_g^{ij}$ and Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{W_g}$, the variance of weight data can be calculated based on $\mathbf{D}$ which models the importance of the data points.

$$Var(f_r) = \tilde{f}_r^T \mathbf{D} \tilde{f}_r \ , \tag{3}$$

where

$$\tilde{f}_r = f_r - \frac{f_r^T \mathbf{D1}}{\mathbf{1}^T \mathbf{D1}} \mathbf{1} \ , \tag{4}$$

Here, we center the data by subtracting the mean from each feature f_r using Equation (4). This is done to prevent a non-zero constant vector such as $\mathbf{1}$ to be assigned a zero Laplacian score, since such a feature obviously does not contain any information.

For a good feature, the larger w_{ij}, the smaller $(f_{ri} - f_{rj})$, and thus it is easy to see that,

$$\sum_{ij}(f_{ri} - f_{rj})^2 w_g^{ij} = 2 f_r^T \mathbf{L} f_r = 2 \tilde{f}_r^T \mathbf{L} \tilde{f}_r \ , \tag{5}$$

Finally, the Laplacian score of the r-th feature is reduced to

$$SC_{Ls}(f_r) = \frac{\tilde{f}_r^T \mathbf{L} \tilde{f}_r}{\tilde{f}_r^T \mathbf{D} \tilde{f}_r} \ , \tag{6}$$

2) MCFS and MRSF: MCFS and MRSF are learning based feature selection methods that first compute an embedding and then use regression coefficients to rank each feature. In the first step, both methods compute a low dimensional embedding represented by the co-ordinate matrix Y. One simple way in deriving low dimensional embedding is to use the Laplacian Eigenmap (LE) [27], a well known dimensionality reduction method. Denote by $\mathbf{Y} = [y_1, y_2, \ldots, y_n]^T$ and $\hat{y}_i$ as transpose of the

i-th row of $\mathbf{Y}$. The idea common to both MCFS and MRSF is to regress all x_i to $\hat{y}_i$. Their differences are used to determine sparseness constraints. MCFS [9] uses ℓ_1-norm regularization and can be regarded as solving the following problems in sequence:

$$\begin{aligned} Y &= arg \min_{\mathbf{Y}\mathbf{Y}^T=I} tr(\mathbf{Y}\mathbf{L}\mathbf{Y}^T) \\ W &= arg \min_{\mathbf{W}} \|\mathbf{X}\mathbf{W} - \mathbf{Y}\|_2^2 + \alpha\|\mathbf{W}\|_1 \end{aligned} \tag{7}$$

Similarly, MRSF first computes the embedding by Eigen decomposition of the graph Laplacian and then regression is with $\ell_{2,1}$-norm regularization. In other words, MRSF can be regarded as solving the following two problems in sequence:

$$\begin{aligned} Y &= arg \min_{\mathbf{Y}\mathbf{Y}^T=I} tr(\mathbf{Y}\mathbf{L}\mathbf{Y}^T) \\ W &= arg \min_{\mathbf{W}} \|\mathbf{X}\mathbf{W} - \mathbf{Y}\|_2^2 + \alpha\|\mathbf{W}\|_{2,1} \end{aligned} \tag{8}$$

MCFS and MRSF employ different sparseness constraints, i.e., ℓ_1 and $\ell_{2,1}$ respectively, in constructing a transformation matrix which is used for selecting features. Nevertheless, the low dimensional embedding, i.e., $\mathbf{Y}$, is determined in the first step and remains fixed in the subsequent ranking or regression step. As a result the performance of feature selection is largely determined by the effectiveness of graph embedding. However, it would be better to learn a graph structure closely linked with the feature selection process.

3) JELSR [28]: Instead of simply using the graph Laplacian to characterize high dimensional data structure and then performing regression, JELSR (joint embedding learning and sparse regression) unifies embedding/learning and sparse regression steps in constructing a new framework for feature selection :

$$(W, Y) = arg \min_{\mathbf{W},\mathbf{Y}\mathbf{Y}^T=I} tr(\mathbf{Y}\mathbf{L}\mathbf{Y}^T) + \beta(\|\mathbf{X}\mathbf{W} - \mathbf{Y}\|_2^2 + \alpha\|\mathbf{W}\|_{2,1}) \tag{9}$$

where α and β are balance parameters. The objective function in Eq.(9) is convex with respect to $\mathbf{W}$ and $\mathbf{Y}$. As a result, $\mathbf{W}$ and $\mathbf{Y}$ can be updated in an alternative way. As we can see from Eq.(29) in [28], the sparse regression of objective function, i.e. the value of $\mathbf{W}$, also affects the low dimensional embedding, i.e., $\mathbf{Y}$. Alternative methods, such as MCFS and MRSF, simply minimize $tr(\mathbf{Y}\mathbf{L}\mathbf{Y}^T)$. Although JELSR performs better in many cases, the optimal graph embedding in JELSR depends heavily

on the transformed data, without making the best use of the original data and the graph edge weights also not learned by the algorithm. This easily leads to the instability performance, especially when encountering a "bad" transformation matrix.

4) LPP [15]: LPP (locality preserving projection) constructs a graph by incorporating neighborhood information derived from the data. Using the graph Laplacian, a transformation is computed to map the data into a subspace by optimally maintaining the local neighborhood information. LPP optimizes a linear transformation $\mathbf{W}$ according to

$$\begin{aligned} &\min_{\mathbf{W}} \sum_{i,j=1}^{n} \|x_i\mathbf{W} - x_j\mathbf{W}\|^2 w_g^{ij} \\ &s.t. \quad \mathbf{W}^T\mathbf{X}^T\mathbf{D}\mathbf{X}\mathbf{W} = 1 \end{aligned} \tag{10}$$

where w_g^{ij} is the graph edge weight which can be computed by Eq.1 and $\mathbf{D}_{ii} = \sum_j w_g^{ij}$. The basic idea underlying LPP is to find a transformation matrix $\mathbf{W}$, which transforms the high-dimensional data $\mathbf{X}$ into a low-dimensional matrix $\mathbf{X}\mathbf{W}$, so as to maximally preserve the local connectivity structure of $\mathbf{X}$ with $\mathbf{X}\mathbf{W}$. Minimizing (10) ensures that, if x_i and x_j are close, and as a result $x_i\mathbf{W}$ and $x_j\mathbf{W}$ are close too.

As described above, LPP seeks a low-dimensional representation with the purpose of preserving the local geometry in the original data. However, such "locality geometry" is completely determined by the artificially constructed neighborhood graph. As a result, its performance may drop seriously if given a "bad" graph. Therefore, it is better to optimize the graph and learn the transformation simultaneously in a unified objective function.

Our proposed method can be discriminated from the previous methods in the following senses:(1) Our propose method selects features to respect both the global and local manifold structure, while most previous feature selection methods only incorporates the local manifold structure; (2) The local structure in previous methods is based on a k-nearest neighbor graph, while our proposed method learns a hypergraph, which can model high-order neighborhood relationship reflected by the original data. (3) JELSR [28] iteratively performs spectral embedding for clustering and sparse spectral regression for feature selection. However, the local structure itself (i.e. the Laplacian

matrix) is not changed during iterations. Our proposed method can adaptively improve the local structure characterization using hypergraph learning.

3. Hypergraph Learning

In this section, we review the definitions of hypergraphs and hypergraph Laplacian. Then, we present our hypergraph construction and learning method.

3.1. Hypergraph Fundamentals

A hypergraph is defined as a triplet $G_H = (V, E, w)$, where $V = \{1, \ldots, n\}$ is the node index set, E is a set of non-empty subsets of V or hyperedges and w is a weight function which associates a real value with each edge. A hypergraph is a generalization of a graph. Unlike graph edges which consist of pairs of vertices, hyperedges are arbitrarily sized sets of vertices. Each hyperedge e is assigned a positive weight $w(e)$. The degree of a hyperedge e, denoted as $\delta(e)$, is the number of vertices in e. For a vertex $v \in V$, the degree is defined to be $d(v) = \sum_{v \in e, e \in E} w(e)$. The diagonal matrix representations for $\delta(e)$, $d(v)$, $w(e)$ are denoted by $\mathbf{D_e}$, $\mathbf{D_v}$ and $\mathbf{W_H}$, respectively. Examples of a hypergraph are shown in Fig. 2(a). For the hypergraph, the vertex set is $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$, where each vertex represents a sample, and the hyperedge set is $E = \{e_1 = \{v_1, v_2, v_3\}, e_2 = \{v_3, v_4, v_5\}, e_3 = \{v_5, v_6\}\}$. The number of vertices constituting each hyperedge represent the order of the relationship between samples.

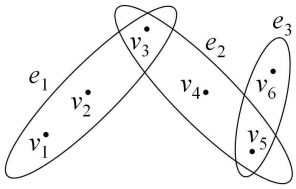
The hypergraph G_H can be represented by a vertex-edge incidence matrix $\mathbf{H} \in R^{|V| \times |E|}$ (see Fig. 2(b)) is defined as follows:

$$h(v, e) = \begin{cases} 1, & \text{if } v \in e \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

According to the definition of $\mathbf{H}$, $d(v) = \sum_{e \in E} w(e) h(v, e)$ and $\delta(e) = \sum_{v \in V} h(v, e)$.

3.2. Hypergraph Laplacian

Although the incidence matrix $\mathbf{H}$ can fully describe the characteristics of a hypergraph, the matrix elements represent vertex-to-hyperedge relationships rather than



(a) Hypergraph Example

$$\boldsymbol{H} = \begin{array}{c} \begin{matrix} e_1 & e_2 & e_3 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{matrix}$$

(b) Incidence Matrix

Figure 2: An Example of Hypergraph.

vertex-to-vertex relationships. To obtain a vertex-to-vertex representation, we need to establish the adjacency matrix and Laplacian matrix for a hypergraph. To achieve this goal, one possible method is to construct a graph with edges weighted by the quotient of the corresponding hyperedge weight and cardinality, e.g., clique expansion [29] and star expansion [29]. As an alternative, one approach is to adopt a matrix representation determined from the adjacency matrix and the associated Laplacian matrix for a hypergraph, e.g. the normalized Laplacian [21]. In this paper, we adopt the method proposed in [21] to build the hypergraph Laplacian. Specifically, the normalized Laplacian matrix of a hypergraph is defined as $\hat{\mathbf{L}}_{\mathbf{H}} = \mathbf{I}_{|\mathbf{V}|} - \mathbf{D}_{\mathbf{v}}^{-\frac{\mathbf{1}}{\mathbf{2}}}\mathbf{H}\mathbf{W}_{\mathbf{H}}\mathbf{D}_{\mathbf{e}}^{-\mathbf{1}}\mathbf{H}^{\mathbf{T}}\mathbf{D}_{\mathbf{v}}^{-\frac{\mathbf{1}}{\mathbf{2}}}$, where $\mathbf{D_v}$ is the diagonal vertex degree matrix whose diagonal element $d(v_i)$ is the summation of the i-th row of $\mathbf{H}$, and $\mathbf{D_e}$ is the diagonal edge degree matrix whose diagonal element $\delta(e_j)$ is the summation of the j-th column of $\mathbf{H}$.

3.3. Hypergraph Construction and Learning

For our hypergraph construction, we regard each sample in the data set as a vertex on hypergraph $G_H = (V, E, w)$, where $V = \{x_1, x_2, \ldots, x_n\}$ is the vertice set. Inspired by the recent developments on sparse representation and ℓ_1-regularized models [30], we propose to generate hyperedges by linking correlated samples. Specifically, each sample can be regarded as a response vector, and can be estimated by a linear

combination of remaining $n-1$ samples, i.e.,

$$x_i = P_i\alpha_i + \varepsilon_i, i = 1, 2, \ldots, n \tag{12}$$

where $P_i = [x_1, x_2, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n]$ denotes a data set including all the samples except the i-th sample (we put 0 in its location), and α_i essentially contains the combination coefficients for different samples in approximating x_i, and $\varepsilon_i \in \Re^n$ is a noise term. A natural method for determining sparse solutions of α_i is formed by solving the following problem:

$$\min_{\alpha_i} \|x_i - P_i\alpha_i\|_2 + \lambda\|\alpha_i\|_1 \tag{13}$$

where $\lambda > 0$ is a regularization parameter controlling the sparsity of α_i. Due to the nature of the ℓ_1-norm penalty, some coefficients will be shrunk to zero if λ is large enough. In this case, we can generate a hyperedge containing the most correlated samples (corresponding to the non-zero coefficients in α_i) with respect to x_i. Different λ values correspond to different sparsity solutions. So instead of generating a single hyperedge for each sample x_i, we generate a group of hyperedges by varying the value of λ over a specified range. Specifically, in our experiments, we vary λ from 0.1 to 0.9 with an incremental step of 0.1.

With this hypergraph construction approach, a large set of remaining hyperedges are generated with redundancy. In addition, they have varying effects on the classification. For example, several hyperedges that are generated from samples close to the classification boundary and they link samples from different classes. Therefore, an effective method for modulating the effects of different hyperedges is needed, such that the weights of redundant hyperedges will be 0, and allowing to select the effective hyperedges.

The importance of preserving local geometric data structure has been well recognized in the recent literature on dimensionality reduction [31] [32] [15] [33]. The local geometric structure of data refers to the local neighborhood relationships for a set of a dataset, which can be characterized through the k nearest neighbors of each sample. By evoking by the principle that nearby points should have similar properties, we define a

regularizer on the hypergraph:

$$\begin{aligned}\Omega = & \frac{1}{2}\sum_{e\in E}\sum_{x_i,x_j\in V}\frac{w(e)h(x_i,e)h(x_j,e)}{\delta(e)}\times(x_i\mathbf{S}-x_j\mathbf{S})^2 \\ = & \mathbf{S^T X^T \hat{L}_H X S} \\ = & \mathbf{S^T X^T}(\mathbf{I}_{|\mathbf{V}|}-\mathbf{D_v^{\frac{-1}{2}} H W_H D_e^{-1} H^T D_v^{\frac{-1}{2}}})\mathbf{X S} \end{aligned} \tag{14}$$

where $\mathbf{S}$ is a linear transformation matrix. The weight of the hyperedge e is assigned a term $\frac{1}{2\delta(e)}\sum_{x_i,x_j\in V(e)}(x_i\mathbf{S}-x_j\mathbf{S})^2$. Here, $V(e)$ is used to denote the set of vertices connected to hyperedge e. As a result, this term measures the feature smoothness on the samples in $V(e)$. Intuitively, hyperedges connecting to the samples from the same class are informative by minimizing (14) with respect to $\mathbf{W_H}$. We ensure that, if x_i and x_j are close, then $x_i\mathbf{S}$ and $x_j\mathbf{S}$ will also be close. Therefore, we use the following objective function to learn the weights of the hyperedges W_H

$$\begin{aligned}&\min_{\mathbf{W_H}} \mathbf{tr}(\mathbf{S^T X^T \hat{L}_H X S}) + \gamma\|\text{diag}(\mathbf{W_H})\|^{\mathbf{2}} \\ &s.t. \quad \sum_{j=1}^{m} W_H^j = 1, W_H^j \geq 0, j = 1,\ldots,m\end{aligned} \tag{15}$$

where m is the number of hyperedges and $\text{diag}(\mathbf{W_H})$ indicates the diagonal vector of $\mathbf{W_H}$, i.e., $(W_H^1, W_H^2, \ldots, W_H^m)$. In order to control the model complexity motivated by the success of sparse learning, we add two constraints $\sum_{j=1}^{m} W_H^j = 1$ and $W_H^j \geq 0$ in (15). In particular, the first constraint fixes the summation of the weights. The second constraint avoids negative weights. Thus, we can see that the solution of $\mathbf{W_H}$ is on a simplex and enjoys the property of sparseness, i.e., the weights assigned to redundant hyperedges will be set to 0.

4. Proposed Framework for Feature Selection

Turning our attention to the task of feature selection, we expect that the transformation matrix $\mathbf{S}$ in (15) satisfies the sparsity property for feature selection. More concretely, we expect that only a few elements in $\mathbf{S}$ are nonzero. As a result the corresponding features $\mathbf{XS}$ are selected since these features are sufficient to preserve the similarity and local geometrical structure of the original data $\mathbf{X}$. We use an $\ell_{2,1}$-norm

regularizer to enforce row sparsity of $\mathbf{S}$, and thus has the effect of feature selection and helps to avoid selecting redundant features. This paper introduces a novel feature selection framework: joint hypergraph learning and sparse regression (referred to as JHLSR). Rather than simply targeting the locality preserving power characterized by hypergraph learning, our proposed model also accommodate the sample similarity structure which can be computed using a predefined similarity measure. In order fulfill this goal, we propose to unify hypergraph learning and sample similarity preserving in forming a new framework as

$$\begin{aligned} &\min_{\mathbf{S},\mathbf{W_H}} \|(\mathbf{AXS})(\mathbf{AXS})^{\mathbf{T}} - \mathbf{K}\|_{\mathbf{F}}^{\mathbf{2}} + \mu \mathbf{tr}(\mathbf{S}^{\mathbf{T}}\mathbf{X}^{\mathbf{T}}\hat{\mathbf{L}}_{\mathbf{H}}\mathbf{XS}) + \lambda\|\mathbf{S}\|_{\mathbf{2,1}} + \gamma\|\text{diag}(\mathbf{W_H})\|^{\mathbf{2}} \\ &s.t. \quad \sum_{j=1}^{m} W_H^j = 1, W_H^j \geq 0, j = 1, \ldots, m \end{aligned} \tag{16}$$

where $\mathbf{A} \in \Re^{l\times n}$ is a binary selection matrix and $\mathbf{K}$ is a predefined similarity matrix. It selects the labeled data out of all data $\mathbf{X}$ when both labeled and unlabeled data are available. $\mathbf{S} \in \Re^{d\times k}$ where d is the number of features in $\mathbf{X}$ and k denotes the dimensions of the transformed data. $\|\cdot\|_F$ denotes the Frobenius matrix norm and $\|\cdot\|_{2,1}$ is the $\ell_{2,1}$-norm of $\mathbf{S}$. The first term in (16) stands for the global structure preservation by emphasizing the pairwise sample similarity, while the second term exploits the local geometric structure of data. The third term is the $\ell_{2,1}$-norm regularization term, which is added to promote row-sparsity. The last term is the diagonal vector of hyperedge weight $\mathbf{W_H}$ and enjoys the sparse property, i.e., the weights of useless hyperedges will be set to 0. To be more specific, the first term aims to select k ($k < d$) features, based on which best preserves the sample similarity as specified by a predefined similarity matrix $\mathbf{K}$. Here, $\mathbf{K}$ is constructed using the Fisher Kernel in supervised learning [2] and by a Gaussian Kernel in unsupervised learning. However, $\|(\mathbf{AXS})(\mathbf{AXS})^{\mathbf{T}} - \mathbf{K}\|_{\mathbf{F}}^{\mathbf{2}}$ is not convex with respect to $\mathbf{S}$. To solve this problem, the method in [2] addresses the following convex optimization problem instead:

$$\min_{\mathbf{S}} \|\mathbf{AXS} - \mathbf{\Phi}\|_{\mathbf{F}}^{\mathbf{2}} + \lambda\|\mathbf{S}\|_{\mathbf{2,1}} \tag{17}$$

where Φ is obtained by decomposing $\mathbf{K}$ as $\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}^{\mathbf{T}}$. Note that $\|\mathbf{S}\|_{2,1}$ is convex. Nevertheless, its derivative does not exist when $\hat{s}_i = 0$ for $i = 1, 2, \ldots, d$. Therefore,

we use the definition $tr(\mathbf{S^T U S}) = \|\mathbf{S}\|_{2,1}/2$ in [28] when $\hat{s}_i$ is not equal to 0. The $\mathbf{U} \in \Re^{d\times d}$ is diagonal with i-th diagonal element where

$$U_{ii} = \frac{1}{2\|\hat{s}_i\|_2} \tag{18}$$

Based on the definitions in (17) and (18), our proposed objective function (16) can be rewritten as

$$\begin{aligned} &\min_{\mathbf{S},\mathbf{W_H}} \|\mathbf{AXS} - \mathbf{\Phi}\|_{\mathbf{F}}^{\mathbf{2}} + \mu\mathbf{tr}(\mathbf{S^T X^T \hat{L}_H X S}) + \lambda\mathbf{tr}(\mathbf{S^T U S}) + \gamma\|\mathrm{diag}(\mathbf{W_H})\|^{\mathbf{2}} \\ &s.t. \quad \sum_{j=1}^{m} W_H^j = 1, W_H^j \geq 0, j = 1, \ldots, m \end{aligned} \tag{19}$$

From (19), it is clear that the proposed objective function has a regularizer on the hyperedge weights and simultaneously optimizes both the transformation matrix $\mathbf{S}$ and the hyperedge weights $\mathbf{W_H}$. In this way, the effects of different hyperedges can be adaptively regulated'. For those hyperedges that are informative, higher weights will be assigned. In addition, our method sparsifies the transformation matrix $\mathbf{S}$, i.e., it optimizes $\mathbf{S}$ by maximally preserving both the local geometrical structure of the data characterized by $\hat{\mathbf{L}}_\mathbf{H}$ and the sample similarity of the labeled data characterized by $\mathbf{K}$.

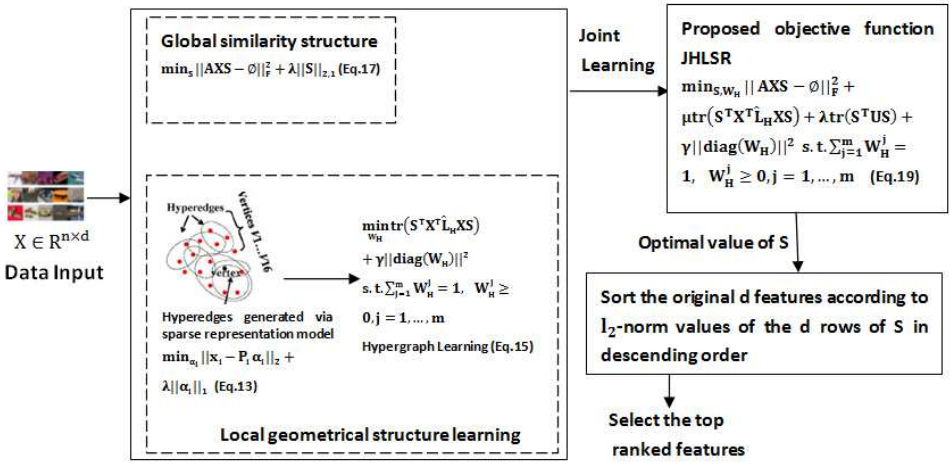


Figure 3: Flowchart of the proposed method

Fig.3 shows the flowchart of the proposed method for feature selection. We pro-

pose a global and local structure preservation framework for feature selection which integrates both global sample similarity structure and local geometrical structure to conduct feature selection (see Eq.19). Concretely, global discriminative information in our framework is preserved by exploiting the underlying sample similarity (see Eq.17). The sample similarity measure may introduce the discriminative information when the data labels are known. Local geometrical structure of data refers to the local neighborhood relationship of a dataset, which can be captured by the results of hypergraph learning (see Eq.15). Specifically, a hypergraph is constructed using each sample as a node, and a hyperedge includes a sample and its correlated samples, with the corresponding non-zero elements extracted in the sparse vector (see Eq.13).

5. Optimization Algorithm

The initial value for each hyperedge weight is set according to the rules given in [34]. First, the $|V| \times |V|$ affinity matrix $\mathbf{A}$ is calculated according to $A_{ij} = \exp\big(-\frac{\|v_i - v_j\|^2}{\sigma^2}\big)$ where σ is the average distance among all vertices. Then, the initial weight for each hyperedge is $W_H^i = \sum_{v_j \in e_i} A_{ij}$. To obtain the global minimal solution of (19), we need an iterative and interleaved optimization process, which can be summarized as in Algorithm 1. In each iteration step, the sparse matrix $\mathbf{S}$ is calculated with the current value $\mathbf{W_H}$, as in equation (21). The diagonal matrix $\mathbf{W_H}$ is updated based on the merely calculated value of $\mathbf{S}$ as in equation (27). After obtaining $\mathbf{W_H}$, we then update the normalized Laplacian matrix $\hat{\mathbf{L}}_\mathbf{H}$ in (23).

We first fix $\mathbf{W_H}$ and solve for $\mathbf{S}$. In other words, we need to solve the following subproblem:

$$\min_{\mathbf{S}} \|\mathbf{AXS} - \mathbf{\Phi}\|_\mathbf{F}^\mathbf{2} + \mu \mathbf{tr}(\mathbf{S^T X^T \hat{L}_H X S}) + \lambda \mathbf{tr}(\mathbf{S^T U S}) \tag{20}$$

Taking the derivative with respect to $\mathbf{S}$ and setting it to zero, we have

$$\frac{\partial}{\partial \mathbf{S}}\Big[\|\mathbf{AXS}-\mathbf{\Phi}\|_F^2+\mu\mathbf{tr}(\mathbf{S^T X^T \hat{L}_H XS})+\lambda\mathbf{tr}(\mathbf{S^T US})\Big]=0$$

$$\begin{cases} \frac{\partial}{\partial \mathbf{S}}\|\mathbf{AXS}-\mathbf{\Phi}\|_F^2=\mathbf{2}(\mathbf{X^T A^T AX})\mathbf{S}-\mathbf{2A^T X^T \Phi}, \\ \\ \frac{\partial}{\partial \mathbf{S}}\mathbf{tr}(\mathbf{S^T US})=\mathbf{2US}, \\ \\ \frac{\partial}{\partial \mathbf{S}}\mathbf{tr}(\mathbf{S^T X^T \hat{L}_H XS})=\mathbf{2}(\mathbf{X^T \hat{L}_H X})\mathbf{S}. \end{cases} \tag{21}$$

$$\mathbf{S}=\Big(\mathbf{X^T}(\mathbf{A^T A}+\mu\mathbf{\hat{L}_H})\mathbf{X}+\lambda\mathbf{U}\Big)^{\mathbf{-1}}\mathbf{A^T X^T \Phi}$$

We then fix $\mathbf{S}$ and solve for $\mathbf{W_H}$. The subproblem becomes

$$\min_{\mathbf{W_H}} \mu\mathbf{tr}(\mathbf{S^T X^T \hat{L}_H XS})+\gamma\|\text{diag}(\mathbf{W_H})\|^{\mathbf{2}} \tag{22}$$

Let

$$\mathbf{\hat{L}_H}=\mathbf{I}_{|\mathbf{V}|}-\mathbf{D_v^{-\frac{1}{2}} H W_H D_e^{-1} H^T D_v^{-\frac{1}{2}}} \tag{23}$$

Then solving the minimization problem in Eq.(22) with respect to $\mathbf{W_H}$ is equivalent to the following problem,

$$\begin{aligned} &\min_{\mathbf{W_H}}\Big\{-\mu\mathbf{tr}(\mathbf{S^T X^T D_v^{\frac{-1}{2}} H W_H D_e^{-1} H^T D_v^{\frac{-1}{2}} XS})+\gamma\|\text{diag}(\mathbf{W_H})\|^{\mathbf{2}}\Big\} \\ &s.t. \quad \sum_{j=1}^{m} W_H^j=1, W_H^j\geq 0, j=1,\ldots,m \end{aligned} \tag{24}$$

Since $\mathbf{W_H}$ and $\mathbf{D_e^{-1}}$ are both diagonal matrices, we let $\mathbf{R}=\mathbf{S^T X^T D_v^{\frac{-1}{2}} H}$ where $\mathbf{R}$ is the matrix $[r_1^T,\ldots,r_m^T]^T$ and $r_i=[r_i^1,r_i^2,\ldots,r_i^m]$. The first term appearing in Eq.(24) can be written as

$$tr(\mathbf{S^T X^T D_v^{\frac{-1}{2}} H W_H D_e^{-1} H^T D_v^{\frac{-1}{2}} XS})=\mathbf{tr}(\mathbf{R W_H D_e^{-1} R^T}) \tag{25}$$

In Eq.(25), its matrix form becomes

$$tr(\mathbf{R}\mathbf{W}_\mathbf{H}\mathbf{D}_\mathbf{e}^{-\mathbf{1}}\mathbf{R}^\mathbf{T}) = tr(\mathbf{R} * \begin{bmatrix} W_H^1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & W_H^m \end{bmatrix} * \begin{bmatrix} \delta(e_1)^{-1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \delta(e_m)^{-1} \end{bmatrix} * R^T)$$

$$= W_H^1\Big(\sum_{i=1}^{m}(r_i^1)^2\Big)\delta(e_1)^{-1} + \cdots + W_H^m\Big(\sum_{i=1}^{m}(r_i^m)^2\Big)\delta(e_m)^{-1}$$

Therefore, the minimization problem in Eq.(24) can be rewritten as

$$\begin{aligned} &\min_{W_H}\Bigg\{ -\mu\Big(W_H^1\Big(\sum_{i=1}^{m}(r_i^1)^2\Big)\delta(e_1)^{-1} + \cdots + W_H^m\Big(\sum_{i=1}^{m}(r_i^m)^2\Big) * \delta(e_m)^{-1}\Big) \\ &+ \gamma\|\text{diag}(W_H)\|^2\Bigg\} \\ &s.t. \quad \sum_{j=1}^{m} W_H^j = 1, W_H^j \geq 0, j = 1, \ldots, m \end{aligned} \tag{26}$$

We use the coordinate descent algorithm to solve the above minimization problem. At each iteration, two elements are selected for updating, and the remainder are fixed. For example, in an iteration, the p-th and the q-th elements, i.e., W_H^p and W_H^q, are selected. According to constrain $\sum_{j=1}^{m} W_H^j = 1$, the summation of W_H^p and W_H^q will not change after this iteration step. Hence, we have

$$\begin{cases} W_H^{p*} = 0, W_H^{q*} = W_H^p + W_H^q, & \text{if} \quad 2\frac{\gamma}{\mu}(W_H^p + W_H^q) \\ & \quad +(S_q - S_p) \leq 0 \\ W_H^{p*} = W_H^p + W_H^q, W_H^{q*} = 0, & \text{if} \quad 2\frac{\gamma}{\mu}(W_H^p + W_H^q) \\ & \quad +(S_p - S_q) \leq 0 \\ W_H^{p*} = \frac{(2\gamma/\mu)(W_H^p + W_H^q) + (S_q - S_p)}{4\gamma/\mu}, & \text{else} \\ W_H^{q*} = W_H^p + W_H^q - W_H^{p*} \end{cases} \tag{27}$$

where $S_p = -\big(\sum_{i=1}^{m}(r_i^p)^2\big) * \delta(e_p)^{-1}$ and $S_q = -\big(\sum_{i=1}^{m}(r_i^q)^2\big) * \delta(e_q)^{-1}$. Note that, in the first line of Eq.(27), we can see that W_H^{p*} will be set to 0. This indicates the solution of W_H has the potential to be sparse, i.e., redundant hyperedges will be removed.

After the optimal value of $\mathbf{S}$ is obtained, we then sort the original d features according to ℓ_2-norm values of the d rows of $\mathbf{S}$ in descending order, and then select the top ranked features.

Algorithm 1: Joint Hypergraph Learning and Sparse Regression (JHLSR)

Input: $\mathbf{X}$,$\mathbf{K}$,$\mathbf{A}$ and regularization parameter μ,λ and γ. $\mathbf{W_H}$ with initial values, hypergraph normalized Laplacian $\hat{\mathbf{L}}_\mathbf{H}$, the matrices $\mathbf{D_v}$,$\mathbf{D_e}$ and $\mathbf{H}$ accordingly.

Output: the otpimal $\mathbf{W_H}$ and sparse matrix $\mathbf{S}$

Step 1: Sparse matrix $\mathbf{S}$ update. ;

repeat

compute S^{t_1+1} by Eq.21;

calculate the diagonal matrix U^{t_1+1}, where the i-th diagonal element is $\frac{1}{2\|\hat{s}_i^{t_1+1}\|_2}$;

$t_1 = t_1 + 1$;

until convergence;

Step 2: $\mathbf{W_H}$ update. Update the weights $\mathbf{W_H}$ with the iterative coordinate descent method introduced in (27) ;

Step 3: $\hat{\mathbf{L}}_\mathbf{H}$ update. Update the normalized Laplacian matrix $\hat{\mathbf{L}}_\mathbf{H}$ in (23) accordingly ;

Step 4: Let $t_2 = t_2 + 1$. if $t_2 > T$, quit iteration and output the results, otherwise go to **Step 1**.

6. Convergence and Complexity Analysis

In this section, we will analyze the properties of the JHLSR algorithm according to three criteria. We first provide the convergence analysis and then discuss computational complexity and parameter determination problems.

6.1. Convergence Proof

Since we have solve JHLSR in an alternative way, we would like to show its convergence behavior. The convergence of **Algorithm 1** can be guaranteed if the following

properties be satisfied.

Theorem 1: The iterative procedure, i.e., **Step 1** in **Algorithm 1**, will monotonically decrease the objective function value in Eq.20.

Theorem 2: When $\mathbf{S}$ is fixed, **Step 2** in **Algorithm 1** will monotonically decrease the objective function value in Eq.19.

Proofs: The proof of Theorem 1 and Theorem 2 are provided in the Appendix A and Appendix B respectively.

From Theorem 1 and Theorem 2, we can see that the iterative procedure in **Algorithm 1** will monotonically decrease the objective function and converge to a global optimum. The following experiments also confirm that the proposed method converges rapidly, typically with a number of iterations is less than 4.

6.2. Complexity Analysis

At each iteration, the main computation of **Step 1** in **Algorithm 1** is to solve the $d \times d$ matrix inverse problem in Eq.21. For many feature selection tasks, the feature dimensionality d is much larger than the number of samples n. The inverse of a large matrix can considerably increase the computational cost. According to [5], we have the following identity:

$$\Big(\mathbf{X^T}(\mathbf{A^T A} + \mu\hat{\mathbf{L}}_\mathbf{H})\mathbf{X} + \lambda\mathbf{U}\Big)^{-\mathbf{1}}\mathbf{X^T} = \mathbf{\Omega X^T}\Big((\mathbf{A^T A} + \mu\hat{\mathbf{L}}_\mathbf{H})\mathbf{X\Omega X^T} + \mathbf{I}\Big)^{-\mathbf{1}} \quad (28)$$

where $\Omega = \frac{1}{\lambda}U^{-1}$ and I is an $n \times n$ identity matrix. From Eq.28, we can convert a $d \times d$ matrix inverse problem to an $n \times n$ one. In doing so, the time complexity of **Step 1** in **Algorithm 1** at each iteration is $O\big(\min(n,d)^3\big)$. And the computational cost of **Step 2** is $O(m^2)$, where m is the number of hyperedges. The computational cost of the hypergraph construction process in Eq.13 is $O(r^3 + n^2)$, where r is the number of nonzero coefficients in α. Thus, the computational complexity of **Algorithm 1** is $\max\Big\{O\big(\min(n,d)^3\big), O(m^2), O(r^3+n^2)\Big\}$.

6.3. Parameter Determination

A parallel issue to optimizing the JHLSR algorithm is selecting optimal values of the parameters μ, λ and γ. The parameter λ and γ are regularization parameters

controlling the sparsity of $\mathbf{S}$ and $\mathbf{W_H}$, and the parameter μ is used to trade off the importance of data similarity preservation and local geometric structure preservation. In order to assign an appropriate value of μ, we employ a cross-validation procedure for μ estimation. In addition, another two parameters, i.e., λ and γ are empirically determined by grid search.

7. Experiments and Comparisons

In this section, we discuss the merits and limitations of the proposed feature selection approach, including a convergence analysis, computational complexity, and parameter determination. A comprehensive experimental study on a variety of data sets is conducted in order to compare our feature selection approach with several state-of-the-art methods in supervised, unsupervised, and semi-supervised modes.

7.1. Experimental Setting

From (16), we observe that the $\mathbf{A} \in \Re^{l \times n}$ is a binary selection matrix and it selects the labeled data out of all data $\mathbf{X}$ when both labeled and unlabeled data are available. $\mathbf{A}$ will degenerate to an identity matrix when only with labeled or unlabeled data are available. According to the value of A, the objective function (19) can implement feature selection in supervised, unsupervised and semi-supervised way. Here, we refer to our proposed method in these three modalities as Sup-JHLSR, Un-JHLSR and Semi-JHLSR respectively. The initial value for each hyperedge weight is set according to the rules given in [34].

To demonstrate the effectiveness of the proposed approach, we conduct experiments on 9 benchmark data sets, i.e., a) the Prostate-GE [5], b) malignant glioma (GLIOMA) data set [35], c) SMK-CAN [36], d) COIL-20 [37], e) handwritten digit image data set MNIST [38], f) Caltech256-2000 [39], g) Scene15 [40], h) ORL [7] and i) ALLAML [5]. Table. 2 summarizes the extent and properties of each of the 9 datasets. For each dataset, 50% of samples are randomly selected as training data, and the remaining are treated as test data in both supervised and the unsupervised modalities. In the semi-supervised case, 5% and 40% samples are randomly selected as labeled and

unlabeled data, respectively, and the remaining are used as test data. We repeat this procedure 10 times and obtain 10 random partitions of the original data. The above feature selection algorithms are evaluated on each partition and the averaged results are reported.

Table 2: Summary of 9 benchmark data sets

Data-set	Sample	Features	Classes
Prostate-GE	102	5966	2
GLIOMA	50	4434	4
SMK-CAN	187	19993	2
COIL-20	1440	1024	20
MNIST	2000	784	10
ORL	400	1024	40
Caltech256-2000	2000	21504	20
Scene15	1500	21504	15
ALLAML	72	7129	2

7.2. Experiment setup

In order to explore the discriminative capabilities of the information captured by our method, we use the selected features for the purpose of classification. We compare the classification results from our proposed method (Sup-JHLSR, Un-JHLSR and Semi-JHLSR) with twelve representative feature selection algorithms.

For supervised learning, six alternative feature selection algorithms are selected as baselines. Compared with our proposed method Sup-JHLSR, most of these methods focus on selecting features that preserve the sample similarity, and neglect the local geometric structure of data. We will briefly introduce these methods one by one.

• Fscore[1]: Fisher Score is a classical feature selection algorithm. It conducts feature selection by evaluating the importance of features one by one. In contract to LapScore and SPEC, Fscore is supervised with class label.

• SPFS [2]: The basic idea of SPFS is to pursue a transformation matrix, which

transform the high-dimensional data to a low-dimensional data, to maximally preserve the global similarity structure of original data.

• mRMR [3]: mRMR is a mutual information based method which is designed to select features that have the maximal statistical dependency on the classification variable, while simultaneously minimizing the redundancy among the selected features.

• LLFS [4]: LLFS selects features which best preserve the global similarity structure of the original data.

• L21RFS [5]: L21RFS shares the spirit of similarity preservation is SPFS. The major difference between L21RFS and SPFS is that the regression loss in SPFS is measured by the Frobenius norm, while the $\ell_{2,1}$-norm is adopted in L21RFS.

• Trace ratio [6]: The trace ratio criterion locates a feature subset for which the within class pairwise affinities are large, while the between class separation is large.

For unsupervised learning, four alternative feature selection algorithms are selected as baselines. A commonly used criterion in these alternative methods is to select the features which best preserve the manifold structure derived from the Laplacian of a graph, where the graph is constructed before hand. However, they separate the processes of learning the graph and feature ranking. In practice, the ideal graph is difficult to define in advance. Because one needs to assign appropriate values for parameters such as the neighborhood size or the heat kernel parameter involved in graph construction, the process is conducted independently of subsequent feature selection. As a result the performance of feature selection is largely determined by the effectiveness of graph construction. Our proposed method Un-JHLSR performs data manifold structure learning and feature selection simultaneously.The structures are adaptively learned from the results of hypergraph learning, and the informative features are selected to preserve the refined structures of data.

• LapScore [7]: LapScore selects features which can best preserve the locality relationship revealed by weight matrix of a predefined graph.

• SPEC [8]: SPEC is a framework for feature selection based on spectral graph theory. It firstly constructs a normalized graph Laplacian and then defines different metrics to measure the importance of each feature. SPEC also can be regarded as an extension of LapScore which is more robust to noise.

• MCFS [9]: Multi-Cluster Feature Selection (MCFS) selects features by sequentially conducting manifold learning and spectral regression.

• JELSR [10]: which joint embedding learning with sparse regression to perform feature selection.

We also compare our obtained results with two state-of-art semi-supervised feature selection methods:

• LSDF [12]: Locality sensitive semi-supervised feature selection (LSDF) is a semi-supervised feature selection approach based on within-class and between-class graph construction.

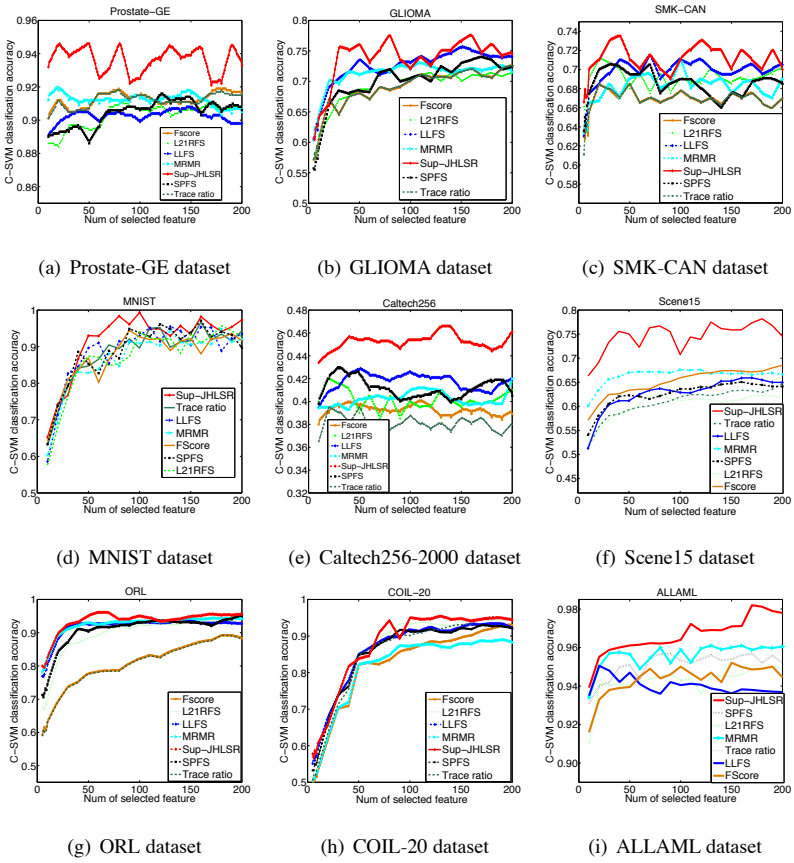
• TRCFS [14]. Noise insensitive trace ratio criterion for feature selection (TRCFS) is a recent semi-supervised algorithm based on noise insensitive trace ratio criterion.

A 10-fold cross-validation strategy using the C-Support Vector Machine (C-SVM) [41] is employed to evaluate the classification performance. We perform the cross-validation on the test samples taken from the feature selection process. Specifically, the entire sample is randomly partitioned into 10 subsets and then we choose one subset for test and use the remaining 9 for training, and this procedure is repeated 10 times. The final accuracy is computed by averaging the accuracies from each of the random subsets.

7.3. Classification Evaluation

Each subfigure shows the classification accuracy versus the number of selected features for each dataset in turn.

1)Results for the Supervised Case (Sup-JHLSR): The classification accuracies obtained with different feature subsets based on supervised learning are shown in Fig.4. From the figure, it is clear that our proposed method Sup-JHLSR is, by and large, superior to the alternative supervised feature selection methods on all the 9 benchmark datasets. Following [2], Table. 3 reports the "aggregated " SVM classification accuracy of different algorithms on each data set. The aggregated SVM classification accuracy is obtained by averaging the averaged accuracy achieved by SVM using the top 10,20,. . .,200 features selected by each algorithm. The boldfaced values are the highest ones.



(a) Prostate-GE dataset (b) GLIOMA dataset (c) SMK-CAN dataset

(d) MNIST dataset (e) Caltech256-2000 dataset (f) Scene15 dataset

(g) ORL dataset (h) COIL-20 dataset (i) ALLAML dataset

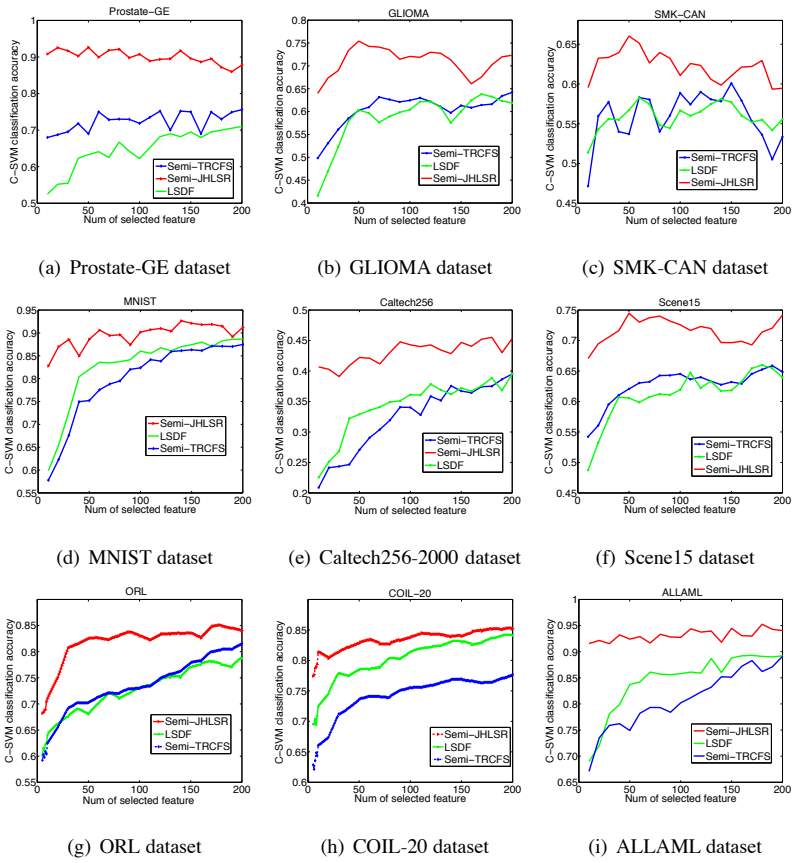
Figure 4: Accuracy rate vs. the number of selected features on 9 benchmark datasets by supervised learning.

The bottom row of Table. 3 shows the averaged classification accuracy for all the algorithms over the 9 datasets. Our method improved the classification accuracy by 5.95% (Fscore), 3.46% (LLFS), 4.77% (L21RFS), 4.06 % (mRMR), 5.59% (Traceratio) and 3.96% (SPFS), respectively, compared to the averaged classification accuracy of all competing methods over the 9 datasets. Meanwhile, our method gives a lower standard deviation and hence more stable than the alternatives. Overall, Fscore gives the worst performance. This may be explained by the fact that it is unable to handle feature redundancy and is prone to select redundant features. SPFS and L21RFS both select a feature subset in which the pairwise similarity between high dimensional sam-

Table 3: Study of supervised cases: aggregated SVM classification accuracy (MEAN ± STD). The last row shows the averaged classification accuracy of all the algorithms over the 9 datasets.

Dataset	Fscore	LLFS	L21RFS	mRMR	Traceratio	SPFS	Sup-JHLSR
Prostate-GE	91.43%	90.29%	90.24%	91.12%	91.43%	90.67%	**93.65**%
	± 4.17	± 3.14	± 4.27	± 4.54	± 3.40	± 4.26	± 3.14
GLIOMA	69.65%	73.17%	70.21%	72.79%	69.65%	70.64%	**74.3**%
	± 2.51	± 2.66	± 2.11	± 2.22	± 2.88	± 2.88	± 2.11
SMK-CAN	67.26%	70.1%	69.16%	68.34 %	67.26%	68.6%	**70.9**%
	± 2.68	± 2.86	± 1.95	± 2.75	± 3.06	± 1.94	± 1.94
MNIST	87.42%	88.53%	85.83%	86.66 %	88.11%	88.7%	**91.55**%
	± 1.92	± 2.05	± 0.78	± 0.94	± 1.88	± 1.65	± 1.26
Caltech256	40.12%	41.6%	39.83%	39.25 %	38.23%	40.18%	**45.69**%
	± 2.29	± 2.09	± 0.95	± 1.59	± 1.65	± 1.12	± 1.03
Scene15	61.6%	61.38%	59.83%	65.2 %	60.85%	61.4%	**74.57**%
	±5.06	± 2.68	± 4.43	± 4.80	± 2.46	± 2.94	± 2.24
ORL	80.3%	90.17%	87.62%	90.34%	80.3%	89.58%	**92.04**%
	± 1.90	± 2.22	± 2.83	± 4.98	± 1.90	± 2.32	± 4.98
COIL-20	84.03%	89.3%	89.86%	83.85 %	89.23%	89.25%	**90.34**%
	± 3.55	± 3.28	± 4.06	±3.28	± 3.30	± 1.72	± 3.06
ALLAML	94.36%	94%	94.25%	95.61 %	94.36%	95.1%	**96.64**%
	± 1.702	± 1.90	± 1.36	± 1.13	± 1.13	± 1.72	± 1.38
AVG	75.13%	77.62%	76.31%	77.02 %	75.49%	77.12%	**81.08**%

ples is maximally preserved. They show inferior performance to our Sup-JHLSR. This indicates that it is important to preserve the sample similarity in identifying discriminative features when the labels of the data are known. From Fig.4 and Table. 3, we observed that those methods which incorporate manifold regularization outperform these methods that do not, i.e., our proposed method Sup-JHLSR is superior to both SPFS and L21RFS in terms of accuracy values for all datasets studied. A possible explanation is that the manifold regularization term causes data space locality information to be preserved in the low dimensional representations. Furthermore, it is demonstrated that the data space geometrical information is crucial for good classification performance.



(a) Prostate-GE dataset (b) GLIOMA dataset (c) SMK-CAN dataset

(d) MNIST dataset (e) Caltech256-2000 dataset (f) Scene15 dataset

(g) ORL dataset (h) COIL-20 dataset (i) ALLAML dataset

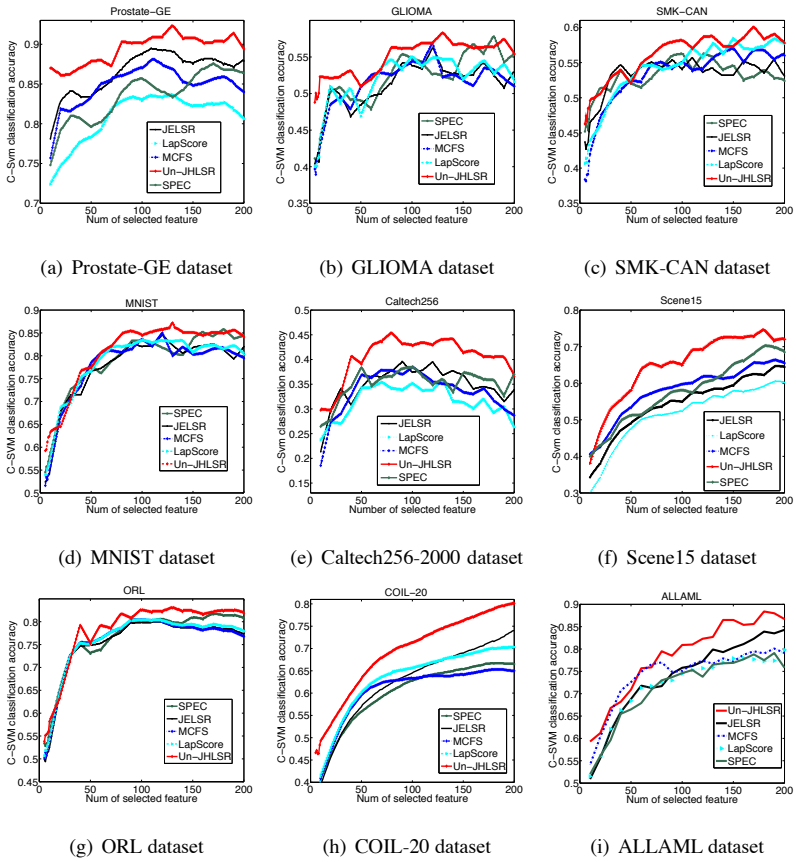
Figure 5: Accuracy rate vs. the number of selected features on 9 benchmark datasets by semi-supervised learning.

2)Results for the Semi-supervised Case (Semi-JHLSR): The classification accuracies for the different feature subsets obtained using semi-supervised learning are shown in Fig.5 and Table. 4. Again, we observe that our proposed method Semi-JHLSR outperforms the alternatives. The aggregated SVM classification accuracy in Table. 4 also clearly shows that the proposed method outperforms each of the competing semi-supervised methods for all datasets studied, and the improvement is in the range from 4.02% to 25.83%. Based on these results, we observe that simultaneously preserving both the sample similarity and the local geometric structure of data is necessary in identifying discriminative features.

Table 4: Study of Semi-supervised cases: aggregated SVM classification accuracy (MEAN $\pm$ STD). The last row shows the averaged classification accuracy of all the algorithms over the 9 datasets.

Dataset	LSDF	Semi-TRCFS	Semi-JHLSR
Prostate-GE	64.2 % $\pm$ 3.06	70.34 % $\pm$ 2.63	$\mathbf{90.03\% \pm 1.37}$
GLIOMA	58.6 % $\pm$ 3.55	60.8 % $\pm$ 3.06	$\mathbf{73.02\% \pm 1.95}$
SMK-CAN	57.4 % $\pm$ 2.68	56.9 % $\pm$ 2.36	$\mathbf{64.23\% \pm 0.97}$
MNIST	80.1 % $\pm$ 2.68	78.37 % $\pm$ 2.78	$\mathbf{89.57\% \pm 1.68}$
Caltech256	34.2 % $\pm$ 3.13	32.5 % $\pm$ 1.94	$\mathbf{43.2\% \pm 2.86}$
Scene15	59.6 % $\pm$ 3.10	62.5 % $\pm$ 3.63	$\mathbf{71.8\% \pm 2.78}$
ORL	73.66 % $\pm$ 3.95	74.28 % $\pm$ 2.33	$\mathbf{81.85\% \pm 0.49}$
COIL-20	78.76 % $\pm$ 2.50	73.26 % $\pm$ 2.82	$\mathbf{82.78\% \pm 0.88}$
ALLAML	84.57 % $\pm$ 3.32	80.89 % $\pm$ 2.78	$\mathbf{93.11\% \pm 4.33}$
AVG	65.68 %	65.54 %	$\mathbf{76.62\%}$

3)Results for the Unsupervised Case (Un-JHLSR): From Fig.6, the proposed method Un-JHLSR still maintains the best classification accuracy on each of the 9 benchmark data sets. The aggregated SVM classification accuracy of different algorithms on each data set is shown in Table. 5. From the results, we draw the following two observations: (1) Firstly, the joint manifold characterization and feature selection methods outperform the methods which separate these two procedures, i.e., Un-JHLSR and JELSR are superior to MCFS and LapScore in terms of accuracy in most cases. (2) Secondly, the proposed method Un-JHLSR shows a significant improvement over the graph based method JELSR. There are three reasons for this improvement in performance. First, The local structure in JELSR is based on a k-nearest neighbor graph, while UN-JHLSR leans a hypergraph. Compared with graph regularization, hypergraph regularization imposes a much stronger constraint on the data samples. Instead of approximating them in terms of pairwise interactions which can lead to a substantial loss of information, the hypergraph representation is effective in capturing the high-order relations among samples. Thus the structural information latent in the data can be effectively preserved. Second, JELSR iteratively performs spectral embedding for



(a) Prostate-GE dataset (b) GLIOMA dataset (c) SMK-CAN dataset

(d) MNIST dataset (e) Caltech256-2000 dataset (f) Scene15 dataset

(g) ORL dataset (h) COIL-20 dataset (i) ALLAML dataset

Figure 6: Accuracy rate vs. the number of selected features on 9 benchmark datasets by unsupervised learning.

clustering and sparse spectral regression for feature selection. However, the local structure itself (i.e. the Laplacian matrix) is not changed during iterations of the algorithm. Our proposed method JHLSR can adaptively improve the local structure by learning the weights of hypergraph. Third, unlike JELSR which only incorporating the local manifold structure, our proposed method JHLSR integrates the merits of local manifold structure and global discriminative sample similarity. Thus, it performs better than the traditional methods.

Taken together, the above experimental results for the supervised, unsupervised, and semi-supervised feature selection modalities demonstrate the effectiveness and ef-

Table 5: Study of Unsupervised cases: aggregated SVM classification accuracy (MEAN $\pm$ STD). The last row shows the averaged classification accuracy of all the algorithms over the 9 datasets.

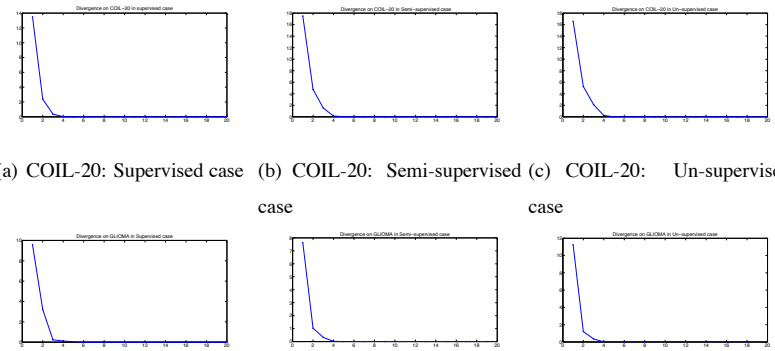
Dataset	SPEC	JELSR	MCFS	LapScore	Un-JHLSR
Prostate-GE	83.32%	86.2%	79.3%	78.4%	**88.04%**
	$\pm$ 2.29	$\pm$ 2.09	$\pm$ 2.19	$\pm$ 1.16	$\pm$ 1.65
GLIOMA	52.75%	51.38%	52.88%	52.65%	**54.78%**
	$\pm$ 3.33	$\pm$ 3.80	$\pm$ 2.27	$\pm$ 2.33	$\pm$ 2.03
SMK-CAN	52.5%	53.7%	50.52%	51.9%	**55.23%**
	$\pm$ 2.25	$\pm$ 3.24	$\pm$ 2.03	$\pm$ 1.82	$\pm$ 3.95
MNIST	78.89%	78.21%	78.93%	78.95%	**80.45%**
	$\pm$ 2.03	$\pm$ 2.21	$\pm$ 2.03	$\pm$ 2.90	$\pm$ 1.70
Caltech256	34.9%	33.7%	35.1%	31.8%	**40.12%**
	$\pm$ 2.50	$\pm$ 3.59	$\pm$ 3.35	$\pm$ 3.95	$\pm$ 2.39
Scene15	57.92%	54.75%	58.45%	50.2%	**66.8%**
	$\pm$ 3.33	$\pm$ 3.04	$\pm$ 2.34	$\pm$ 3.23	$\pm$ 2.52
ORL	72.12%	73.3%	72.84%	72.9%	**76.2%**
	$\pm$ 3.10	$\pm$ 3.13	$\pm$ 1.35	$\pm$ 3.74	$\pm$ 2.60
COIL-20	62.36%	64.93%	65.13%	65.42%	**71.6%**
	$\pm$4.16	$\pm$4.75	$\pm$5.06	$\pm$2.97	$\pm$3.23
ALLAML	70.25%	76.25%	73.4%	71.54%	**81.38%**
	$\pm$ 0.95	$\pm$ 1.12	$\pm$ 2.29	$\pm$ 1.59	$\pm$ 1.03
AVG	62.78%	63.60%	62.95%	61.53%	**68.29%**

ficiency of the proposed JHLSR framework.

7.4. Convergence Results

In this section, we provide some numerical results to illustrate the convergence behavior of our algorithm JHLSR. Two datasets, i.e., COIL-20 and GLIOMA, are employed. Since S is used for feature selection, we would like to measure the variance between two sequential S using the following metric:

$$Error(t) = \|S^{t+1} - S^t\|_2^2 \tag{29}$$



(a) COIL-20: Supervised case (b) COIL-20: Semi-supervised case (c) COIL-20: Un-supervised case

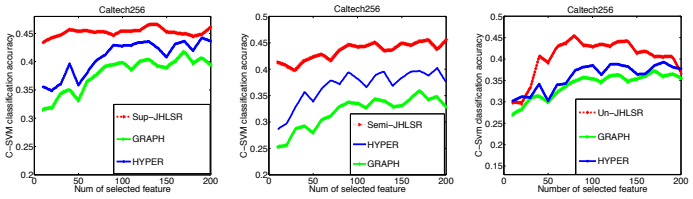
(d) GLIOMA: Supervised case (e) GLIOMA: Semi-supervised case (f) GLIOMA: Un-supervised case

Figure 7: Convergence behavior of JHLSR. There are mainly 20 iterations. x-axis represents the number of iterations and y-axis represents the divergence between two sonsecutive S measure by Eq.29. As observed, JHLSR always converge within 4 iterations.

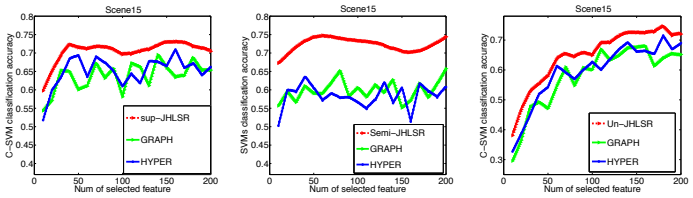
As seen from Fig.7, the divergence between two consecutive S converges to zero, which means that the final results will not be changed drastically. Convergence is fast, requiring less than 4 iterations.

7.5. Effect of Adaptive Structure Learning by Hypergraph

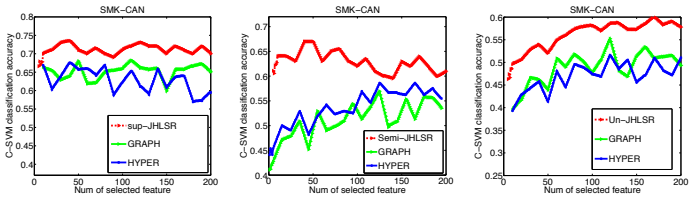
To further illustrate the effectiveness of JHLSR in preserving the local manifold structure of the data, we compare JHLSR with regular hypergraph learning (HYPER) [21] (i.e. with no learning of the hyperedge weights) and a graph based version of the proposed algorithm (referred to as GRAPH). The main experimental results are presented in Fig.8 and a few interesting observations can be made. First, JHLSR consistently outperforms HYPER and GRAPH on all the datasets studied. The main reason is that JHLSR can represent diverse relations among data samples, and adaptively improve the local structure from the results of hypergraph learning. Second, JHLSR consistently performs better than the conventional hypergraph learning algorithm (i.e. HYPER), and this result suggests that the simultaneous learning of hyperedge weights and feature selection is a better strategy. Third, for the supervised case, there are few differences among the three methods, since the label information is more crucial than modeling the local manifold structure of data for the subsequent classification task.



(a) Caltech256: Supervised case (b) Caltech256: Semi-supervised case (c) Caltech256: Un-supervised case



(d) Scene15: Supervised case (e) Scene15: Semi-supervised case (f) Scene15: Un-supervised case



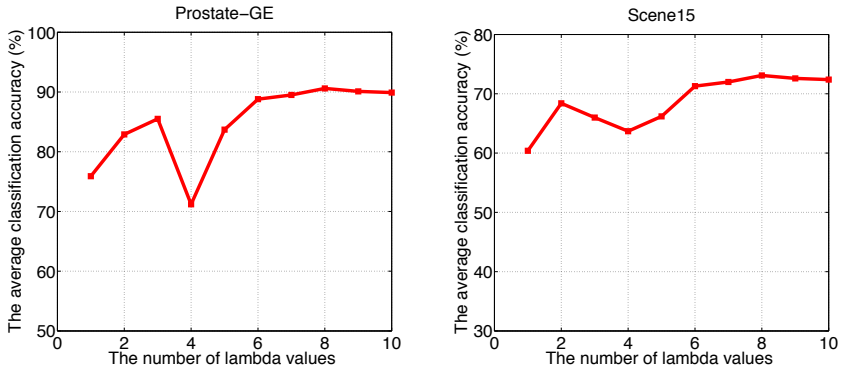
(g) SMK-CAN: Supervised case (h) SMK-CAN: Semi-supervised case (i) SMK-CAN: Un-supervised case

Figure 8: Accuracy rate vs. the number of selected features on three benchmark datasets by different methods (JHLSR,HYPER,GRAPH).

However, in the semi-supervised and unsupervised cases, JHLSR gains a significant improvement over HYPER and GRAPH. This is because when the labeled data are scarce, feature selection aims to select the features that well maintain the underlying local manifold structure. In this case, the local structure itself (i.e. the Laplacian matrix) in HYPER is not changed during iterations of the algorithm. JHLSR can adaptively improve the local manifold structure by updating the hyperedge weights.

7.6. *Effect of Hypergraph Learning*

In this section, we will evaluate the effectiveness of hypergraph construction using the sparse representation model and hyperedge weight learning.

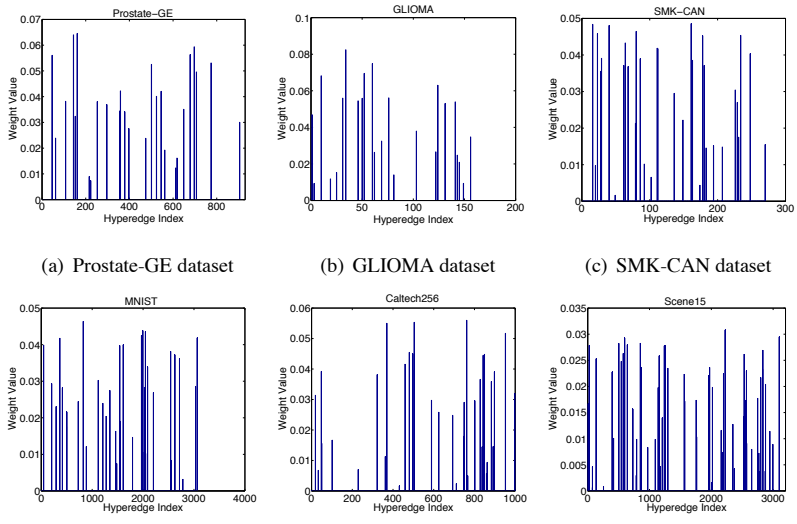


(a) Prostate-GE dataset (b) Scene15 dataset

Figure 9: Classification accuracy w.r.t. the use of different number of λ values in Eq.13.

To investigate the effect of different numbers of λ values (in Eq.13) on the classification performance of the proposed method, we test 10 groups of λ values, i.e., $\{0.1\}$, $\{0.1, 0.2\}$, $\{0.1, 0.2, 0.3\}$, ..., $\{0.1, 0.2, \ldots, 0.9\}$, $\{0.1, 0.2, \ldots, 0.9, 1\}$. Fig.9 gives the classification results. The figure shows that with an increase in the number of λ values, the classification accuracy first increases to a high value and then decreases, finally converges to a highest value and reaches a steady state. This observation verifies that the range $(0.1 : 0.9)$ is enough for λ in Eq.13.

Hypergraph construction using the sparse representation model will produce some redundant hyperedges. Therefore, in order to regulate the effects of different hyperedges, we place a regularizer (i.e. $\sum_{j=1}^{m} W_H^j = 1$ and $W_H^j \geq 0$) on the hyperedge weights. In this way, the effects of different hyperedges can be adaptively regulated and useless or redundant hyperedges can be discarded (i.e., the weights of redundant hyperedges will ideally be 0), and thus, we can select the most effective hyperedges. Fig.10 visualizes the values of the hyperedge weights on the six data sets. It is clear that different hyperedges have different weights and some hyperedge weights are 0. For clear comparison, we illustrate the number of non-zero weight hyperedges dur-



(a) Prostate-GE dataset (b) GLIOMA dataset (c) SMK-CAN dataset

(d) MNIST dataset (e) Caltech256-2000 dataset (f) Scene15 dataset

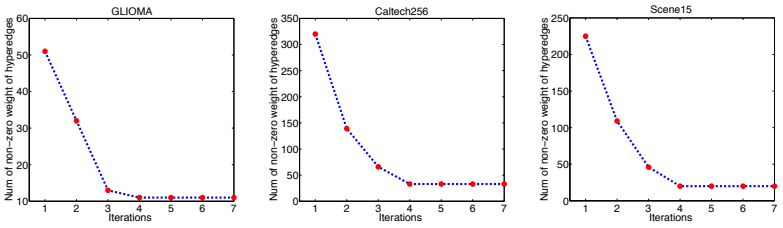
Figure 10: Illustration of the learned hyperedge weights in the proposed method.

ing learning in Fig.11. As demonstrated in Fig.11, the number of non-zero weight hyperedges iteratively decreases until reaching a steady state. Therefore, only a very small number of hyperedges are preserved after learning. The results further verify the effectiveness of hypergraph learning.

8. Conclusion

In this paper, we have proposed a hypergraph learning approach for feature selection, aimed at capturing higher order sample relations in sets of data. The approach not only incorporates a robust hyperedge construction method, but also allows for the simultaneously learning of hyperedge weights and feature selection based on matrix sparsification. The learned hyperedges weight are shown to better characterize the manifold structure of the data. Experimental results for the cases of supervised, unsupervised and semi-supervised feature selection demonstrate both the effectiveness and efficiency of the proposed JHLSR framework.

There are a number of shortcomings of the proposed method, which we aim to address in future work. Firstly the JHLSR method has three parameters that need to be



(a) GLIOMA dataset (b) Caltech256-2000 dataset (c) Scene15 dataset

Figure 11: Illustration of the number of non-zero weight of hyperedges with iterations.

hand-tuned, and which is computationally cumbersome for real world applications. To reduce this burden, we will replace the convex regularizations on S and W_H with ℓ_{20} or ℓ_0 norm. Secondly, both the global and local structures (i.e. K and L_H) in JHLSR are based on all the available features. We will investigate how to refine the estimation of these structures using the selected features.

At a more ambitious level, it would also be interesting to explore whether the semi-supervised approach to feature selection presented here could be cast into the harmonic framework [42]. This would provide a natural way of learning the hyper graph weights in a semi-supervised setting.

Appendix A. Proof of Theorem 1

We prove that the proposed Algorithm 1 makes the values of the objective function in Eq.20 monotonically decrease. We first give a Lemma [5] as follows, which will be used in our proof.

Lemma 1: For any nonzero vectors $\mathbf{a}$,$\mathbf{b} \in \Re^d$, the following result follows:

$$\|\mathbf{a}\|_2 - \frac{\|\mathbf{a}\|_2^2}{2\|\mathbf{b}\|_2} \leq \|\mathbf{b}\|_2 - \frac{\|\mathbf{b}\|_2^2}{2\|\mathbf{b}\|_2} \tag{A.1}$$

Proof: For any nonzero vectors $\mathbf{a}$,$\mathbf{b} \in \Re^d$, there exists

$$\|\mathbf{a}\|_2\|\mathbf{b}\|_2 \leq \frac{1}{2}(\|\mathbf{a}\|_2^2 + \|\mathbf{b}\|_2^2) \tag{A.2}$$

For any $\mathbf{b} \neq 0$, we have

$$\|\mathbf{a}\|_2 \leq \frac{1}{2}\frac{\|\mathbf{a}\|_2^2}{\|\mathbf{b}\|_2} + \frac{1}{2}\frac{\|\mathbf{b}\|_2^2}{\|\mathbf{b}\|_2} \tag{A.3}$$

By Eq.A.3, we obtain

$$\|\mathbf{a}\|_2 - \frac{1}{2}\frac{\|\mathbf{a}\|_2^2}{\|\mathbf{b}\|_2} \leq \frac{1}{2}\frac{\|\mathbf{b}\|_2^2}{\|\mathbf{b}\|_2} \tag{A.4}$$

This completes the proof.

According to [43, 44, 45, 10], optimizing the non-smooth convex form $\|S\|_{2,1}$ can be transferred to iteratively optimize U and S in $tr(S^TUS)$. As seen from the **Step 1** in Algorithm 1, when we fix U as U^{t_1} in the t_1-th iteration and update S^{t_1+1}, Eq.20 can be rewritten as

$$S^{t_1+1} = \arg\min_S \left\{ \|AXS - \Phi\|_F^2 + \mu tr(S^T X^T \hat{L}_H X S) + \lambda tr(S^T U^{t_1} S) \right\} \tag{A.5}$$

and the following inequality holds:

$$\begin{aligned} &\|AXS^{t_1+1} - \Phi\|_F^2 + \mu tr\Big((S^{t_1+1})^T X^T \hat{L}_H X S^{t_1+1}\Big) + \lambda tr\Big((S^{t_1+1})^T U^{t_1} S^{t_1+1}\Big) \\ &\quad \leqslant \|AXS^{t_1} - \Phi\|_F^2 + \mu tr\Big((S^{t_1})^T X^T \hat{L}_H X S^{t_1}\Big) + \lambda tr\Big((S^{t_1})^T U^{t_1} S^{t_1}\Big) \end{aligned} \tag{A.6}$$

Since $U_{ii} = \frac{1}{2\|\hat{s}_i\|_2}$ and the inequality in $\|S\|_{2,1} = \sum_{i=1}^d \|\hat{s}_i\|_2$, then Eq.A.6 can be rewritten as

$$\begin{aligned} &\|AXS^{t_1+1} - \Phi\|_F^2 + \mu tr\Big((S^{t_1+1})^T X^T \hat{L}_H X S^{t_1+1}\Big) + \lambda \sum_{i=1}^d \frac{\|\hat{s}_i^{t_1+1}\|_2^2}{2\|\hat{s}_i^{t_1}\|_2} \\ &\quad \leqslant \|AXS^{t_1} - \Phi\|_F^2 + \mu tr\Big((S^{t_1})^T X^T \hat{L}_H X S^{t_1}\Big) + \lambda \sum_{i=1}^d \frac{\|\hat{s}_i^{t_1}\|_2^2}{2\|\hat{s}_i^{t_1}\|_2} \end{aligned} \tag{A.7}$$

Recalling the result gain in **Lemma 1**, we know that

$$\sum_{i=1}^d \|\hat{s}_i^{t_1+1}\|_2 - \sum_{i=1}^d \frac{\|\hat{s}_i^{t_1+1}\|_2^2}{2\|\hat{s}_i^{t_1}\|_2} \leqslant \sum_{i=1}^d \|\hat{s}_i^{t_1}\|_2 - \sum_{i=1}^d \frac{\|\hat{s}_i^{t_1}\|_2^2}{2\|\hat{s}_i^{t_1}\|_2} \tag{A.8}$$

Based on Eq.A.7 and Eq.A.8, we have the following result:

$$\begin{aligned} &\|AXS^{t_1+1} - \Phi\|_F^2 + \mu tr\Big((S^{t_1+1})^T X^T \hat{L}_H X S^{t_1+1}\Big) + \lambda \|S^{t_1+1}\|_{2,1} \\ &\quad \leqslant \|AXS^{t_1} - \Phi\|_F^2 + \mu tr\Big((S^{t_1})^T X^T \hat{L}_H X S^{t_1}\Big) + \lambda \|S^{t_1}\|_{2,1} \end{aligned} \tag{A.9}$$

This inequality indicates that function in Eq.20 will monotonically decrease in each iteration. Therefore, **Step 1** in Algorithm 1 will converge.

Appendix B. Proof of Theorem 2

Lemma 2: For any nonzero vectors x and y, we attempt to solve the following optimization problem:

$$\begin{aligned} &\min_{x,y} \gamma(x^2+y^2)+ax+by \\ &s.t. \quad x+y=c, x\geq 0, y\geq 0 \end{aligned} \tag{B.1}$$

Hence, we have the optimal solution of the above problem as

$$\begin{cases} x=0, y=c, & \text{if} \quad -\frac{b-a-2\gamma c}{4\gamma}\geq c \\ x=c, y=0, & \text{if} \quad -\frac{b-a-2\gamma c}{4\gamma}\leq 0 \\ x=\frac{2\gamma c+b-a}{4\gamma}, & \text{else} \\ y=c-x \end{cases} \tag{B.2}$$

Proof: Since $x=c-y$, we add it into the objective function Eq.B.1, then we have

$$\min_{y} \gamma(c-y)^2+\gamma y^2+a(c-y)+by \tag{B.3}$$

Rewriting the above optimization problem in Eq.B.3 as

$$\min_{y} \left\{ 2\gamma y^2+(b-a-2\gamma c)y+\gamma c^2+ac \right\} \tag{B.4}$$

the following optimal solutions hold:

$$\begin{cases} y=c, & \text{if} \quad -\frac{b-a-2\gamma c}{4\gamma}\geq c \\ y=0, & \text{if} \quad -\frac{b-a-2\gamma c}{4\gamma}\leq 0 \\ y=-\frac{b-a-2\gamma c}{4\gamma}, & \text{else} \end{cases} \tag{B.5}$$

This completes the proof.

When we fix S and solve for W_H, the objective function in Eq.19 can be rewritten as

$$\min_{W_H} \left\{ -\mu tr(RW_H D_e^{-1} R^T)+\gamma \|\text{diag}(W_H)\|^2 \right\} \tag{B.6}$$

where $R=S^T X^T D_v^{\frac{-1}{2}} H$. The aforementioned problem can be solved using an alternating optimization process. By using a coordinate descent algorithm, we develop an iterative process that alternately updates the sparse matrix S and the weight value

W_H. At each iteration, two elements are selected for updating, whereas the remaining are left fixed. For example, in t_2-th iteration, the p-th and the q-th elements, i.e., W_H^p and W_H^q, are selected. After the iteration, we update W_H^p and W_H^q as W_H^{p*} and W_H^{q*} respectively (see Eq.27).

Recalling the results in **Lemma 2**, we know that if we let $b = \mu S_q$, $a = \mu S_p$, $x = W_H^p$ and $y = W_H^q$, we have

$$(W_H^{p*}, W_H^{q*}) = \min_{W_H^p, W_H^q} -\mu\Big(W_H^p\Big(\sum_{i=1}^{m}(r_i^p)^2\Big)\delta(e_p)^{-1} + W_H^q\Big(\sum_{i=1}^{m}(r_i^q)^2\Big)\delta(e_q)^{-1}\Big) + \gamma(W_H^{p2} + W_H^{q2}) \tag{B.7}$$

Therefore, the following inequality holds:

$$\begin{aligned} &-\mu\Big(W_H^{p*}\Big(\sum_{i=1}^{m}(r_i^p)^2\Big)\delta(e_p)^{-1} + W_H^{q*}\Big(\sum_{i=1}^{m}(r_i^q)^2\Big)\delta(e_q)^{-1}\Big) + \gamma(W_H^{p*2} + W_H^{q*2}) \\ &\leqslant -\mu\Big(W_H^{p}\Big(\sum_{i=1}^{m}(r_i^p)^2\Big)\delta(e_p)^{-1} + W_H^{q}\Big(\sum_{i=1}^{m}(r_i^q)^2\Big)\delta(e_q)^{-1}\Big) + \gamma(W_H^{p2} + W_H^{q2}) \end{aligned} \tag{B.8}$$

As seen from Eq.B.8, since each step decreases the objective function, the convergence of the alternating optimization process is guaranteed. As a result, the objective function Eq.19 has a global optimum solution.

Acknowledgment

This work is supported by National Natural Science Foundation of China (Grant No.61402389 and 61503422).

References

[1] R. O. Duda, P. E. Hart, D. G. Stork, Pattern classification, John Wiley & Sons, 2012.

[2] Z. Zhao, L. Wang, H. Liu, J. Ye, On similarity preserving feature selection, IEEE Transactions on Knowledge and Data Engineering 25 (3) (2013) 619–632.

[3] H. Peng, F. Long, C. Ding, Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (8) (2005) 1226–1238.

[4] Y. Sun, S. Todorovic, S. Goodison, Local-learning-based feature selection for high-dimensional data analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (9) (2010) 1610–1626.

[5] 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.

[6] F. Nie, S. Xiang, Y. Jia, C. Zhang, S. Yan, Trace ratio criterion for feature selection., in: AAAI, Vol. 2, 2008, pp. 671–676.

[7] X. He, D. Cai, P. Niyogi, Laplacian score for feature selection, in: Advances in neural information processing systems, 2005, pp. 507–514.

[8] 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.

[9] D. Cai, C. Zhang, X. He, Unsupervised feature selection for multi-cluster data, in: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2010, pp. 333–342.

[10] C. Hou, F. Nie, X. Li, D. Yi, Y. Wu, Joint embedding learning and sparse regression: A framework for unsupervised feature selection, IEEE Transactions on Cybernetics 44 (6) (2014) 793–804.

[11] Z. Xu, I. King, M.-T. Lyu, R. Jin, Discriminative semi-supervised feature selection via manifold regularization, IEEE Transactions on Neural Networks 21 (7) (2010) 1033–1047.

[12] J. Zhao, K. Lu, X. He, Locality sensitive semi-supervised feature selection, Neurocomputing 71 (10) (2008) 1842–1849.