Joint Embedding Learning and Sparse Regression: A Framework for Unsupervised Feature Selection

Chenping Hou, Member, IEEE, Feiping Nie, Xuelong Li, Fellow, IEEE, Dongyun Yi, Yi Wu

Abstract—Feature selection has aroused considerable research interests during the last few decades. Traditional learning based feature selection methods separate embedding learning and feature ranking. In this paper, we propose a novel unsupervised feature selection framework, termed as joint embedding learning and sparse regression (JELSR), in which the embedding learning and sparse regression are jointly performed. In specific, the proposed JELSR joins embedding learning with sparse regression to perform feature selection. To show the effectiveness of the proposed framework, we also provide a method using the weight via local linear approximation and adding the penalty of the proposed framework, we also provide a method using the weight via local linear approximation and adding the $\ell_2,\lambda$-norm regularization, and design an effective algorithm to solve the corresponding optimization problem. Furthermore, we also conduct some insightful discussion on the proposed feature selection, including the convergence analysis, computational complexity and parameter determination. In all, the proposed framework not only provides a new perspective to view traditional methods, but also evokes some other deep researches for feature selection. Compared with traditional unsupervised feature selection methods, our approach could integrate the merits of embedding learning and sparse regression. Promising experimental results on different kinds of data sets, including image, voice data and biological data, have validated the effectiveness of our proposed algorithm.

Index Terms—feature selection, embedding learning, sparse regression, pattern recognition.

I. INTRODUCTION

High dimensional data is commonly confronted in many practical applications such as pattern recognition, computer vision and data mining. It significantly increases the time and space requirements to process the data. To address this problem, dimensionality reduction techniques are proposed to reduce the dimensionality of the high dimensional data by finding a relevant feature subset. It obtains a smaller set of representative features and retains the optimal salient characteristics. Preprocessing data in this way not only decreases the processing time but also leads to more compactness and better generalization of the learned models [1].

In the literature, there are mainly two distinct ways for dimensionality reduction: feature selection (or 'feature ranking') [1], [2], [3], [4], [5], [6], [7], [8] and feature learning (or 'feature extraction') [9], [10], [11], [12]. Feature selection aims at extracting a few relevant features to represent the original data while feature learning combines several original features to form new representations. Compared with feature learning which can create new features, feature selection does not change the original representations of data variables. Additionally, another advantage of feature selection is that we only need to calculate or collect these concerning features once we have determined the selected features.

Consequently, many efforts have been devoted to addressing the problem of feature selection during the last few decades [1], [2], [3], [4], [5], [6], [7], [8]. There are mainly two different kinds of feature selection approaches: supervised and unsupervised. Since there is no label information available in the unsupervised feature selection, it is more difficult than supervised one and there are relatively few investigations dedicated to this topic. Most unsupervised feature selection approaches are either based on filters [13], [14], [15], wrappers [4], [16] or embedding [17], [18], [19]. Jensen and Shen et al have also proposed a lot of promising feature selection approaches based on rough set and fuzzy rough set theory [20], [21]. Although the performances of traditional unsupervised feature selection approaches are prominent in many cases, their efficiencies can also be improved since (1) from the view of manifold learning [22][23], the high dimensional data are nearly lying on a low dimensional manifold. Traditional methods have not taken fully considerations about the manifold structure. (2) Different from feature learning, traditional feature selection approaches only employ data statistical character to rank the features essentially. They are often lack of using the learning mechanism as in feature learning, which is proved to be powerful and widely used in many areas [24] [25].

Recently, to leverage both the manifold structure and learning mechanism, some investigations have emerged. Typical methods include: Pca Score (PcaScor) [26], Laplacian Score (LapScor) [27], Spectral Feature Selection (SPEC) [28], Multi-Cluster Feature Selection (MCFS) [29] and Minimum Redundancy Spectral Feature selection (MRSF) [30]. Usually, these methods use various graphs to characterize the manifold structure at first. LapScor and SPEC then compute different metrics to rank each feature, MCFS and MRSF, however, add sparse constraints in multi-output regression. Compared with the traditional unsupervised feature selection approaches, these methods have proved to perform better in many cases [29], [30]. Nevertheless, their performances can also be further
improved since all the aforementioned methods separate manifold characterization and feature selection. Once the graph is determined to characterize the manifold structure, it is fixed in the following ranking or regression steps. Therefore, the performance of feature selection is largely determined by the effectiveness of graph construction.

In order to fulfill this goal, this paper introduces a novel unsupervised feature selection framework: joint embedding learning and sparse regression (JELSR). Instead of simply using the graph laplacian to characterize high dimensional data’s structure and then regression, we propose to unify embedding learning and sparse regression in forming a new framework. Some popular feature selection methods, such as MCFS and MRSF, can be viewed as special cases within the proposed framework. By using the local linear approximation weights and $\ell_2,1$-norm regularization, we provide a new method, which is also named as JELSR, for illustration. It can learn a sparse transformation matrix for feature selection. We also provide a theoretical discussion to solve the proposed problem. Compared to existing methods, our framework integrates the merits of both manifold learning and sparse regression. Thus, it performs better than traditional methods.

The rest of this paper is organized as follows. The related works are introduced in Section II. In Section III, we propose our framework, formulate JELSR and provide an effective solution algorithm. Section IV presents the deep analyses about the proposed method, including convergence behavior, computational complexity etc. Section V provides some comparing results on various kinds of data sets, followed by the conclusion and future works in Section VI.

II. RELATED WORK

In this section, we would like to introduce some famous learning-based unsupervised feature selection approaches, i.e., PcaScor, LapScor, SPEC, MCFS and MRSF. Take SR as an example for feature learning approaches, we also introduce it in this section. The reason why we choose these approaches is that they are closely related to our algorithm.

A. PcaScor, LapScor, SPEC, MCFS and MRSF

We will briefly introduce these methods one by one.

1) PcaScor: The original PcaScor algorithm is one of the firstly proposed unsupervised feature selection methods. It is based on Principal Component Analysis (PCA). The basic idea is to compute the variance of each feature and rank all features based on the computed variances. We choose the features corresponding to the first $s$ largest variances. Formally, denote $\{x_i = [x_{i1}, x_{i2}, \ldots, x_{id}]^T \in \mathbb{R}^d | i = 1, 2, \ldots, n\}$ as the original data, where $d$ is the dimensionality of original data and $n$ is the number of points. Denote $X = [x_1, x_2, \ldots, x_n] \in \mathbb{R}^{d \times n}$. The covariance matrix of these data is

$$\text{var}(X) = \sum_{i=1}^{n} (x_i - \bar{m})(x_i - \bar{m})^T,$$

where $\bar{m}$ is the sample mean.

PcaScor ranks every dimension of original data based on their variance, i.e., the diagonal elements in $\text{var}(X)$. The larger this value is, the more important this feature is. We select a fixed number of the most important features.

2) LapScor: LapScor is another famous unsupervised feature selection approach. It can be regarded as the variants of PcaScor in various applications. As seen from the above procedure, PcaScor only uses the statistical characteristic of each feature. It does not take fully considerations about the manifold structure of original data.

Besides, there are also some other related researches to extend PCA for feature selection, such as Principle Feature Analysis [31] and the works in [32], [33]. They can be regarded as the variants of PcaScor in various applications. As seen from the above procedure, PcaScor only uses the statistical characteristic of each feature. It does not take fully considerations about the manifold structure of original data.

Step 1. Constructing a $k$-nearest neighborhood graph $G$ with $n$ nodes, where $k$ is the number of nearest neighbors.

Step 2. Computing a weight matrix $S$ on previous graph $G$ by different strategies [27].

Step 3. Removing the mean of each feature. Define $f_i = [X_{i1}, X_{i2}, \ldots, X_{in}]$ as a collection of samples for the $i$-th feature. Denote $D$ as a diagonal matrix whose element $D_{ii} = \sum_{j=1}^{n} S_{ij}$. In other words, $D = \text{diag}(S1)$, where $1$ is a vector of all ones. Denote $L = D - S$, which is known as the graph laplacian [34]. To remove the feature mean, we define

$$\hat{f}_i = f_i - \frac{f_i^T D 1}{1^T D 1}. \quad (2)$$

Step 4. Computing the laplacian score for the $i$-th feature as follows.

$$a_i = \frac{\hat{f}_i^T L \hat{f}_i}{\hat{f}_i^T D \hat{f}_i}. \quad (3)$$

Step 5. Selecting the features. We rank the features based on $\{a_i\}_{i=1}^{d}$ and choose features corresponding to the $s$ largest $a_i$.

As seen from the above deduction, LapScor uses the graph laplacian to characterize the manifold structure of high dimensional data. Nevertheless, it does not take learning mechanism in feature selection.
3) SPEC: SPEC is a framework for feature selection based on spectral graph theory. The authors exploit common properties underlying supervised and unsupervised feature selection algorithms and propose a novel framework for supervised and unsupervised feature selection.

The main procedure of SPEC is first to construct a normalized graph laplacian and then to define different metrics to measure the importance of each feature. Similar to LapScor, it constructs the same graph laplacian as in Step 3 of LapScor and then normalizes it as $\tilde{L} = D^{-1/2}LD^{-1/2}$. Finally, it uses these three different evaluation metrics to rank each feature.

As seen from the above procedure, the graph laplacian in SPEC is only used to characterize data structure. It has not fully reflected the requirement of feature selection.

4) MCFS and MRSF: MCFS and MRSF are two famous learning based feature selection approaches. They all compute the embedding at first and then use regression coefficient to rank each feature. More concretely, in the first step, both methods compute the embedding of $x_i$, denoted as $y_i \in \mathbb{R}^m$, where $m$ is the dimensionality of embedding space. One common way in deriving low dimensional embedding is to use Laplacian Eigenmap (LE)[34], a famous dimensionality reduction approach. Denote $Y = [y_1, y_2, \cdots, y_n]$ and $\tilde{y}_i$ as transpose of the $i$-th row of $Y$. The common idea of MCFS and MRSF is to regress all $x_i$ to $\tilde{y}_i$. Their differences are the concrete forms of sparse constraints. Denote $a_i$ as a $d$-dimensional regression vector. The formulation of MCFS is

$$\min_{a_i} \| \tilde{y}_i - X^T a_i \|^2 + \beta \| a_i \|_1$$  \hspace{1cm} (4)

Clearly, MCFS has added the $\ell_1$-norm regularization in regression. The optimization problem in Eq. (4) is known as Lasso[35], which has been widely investigated in the field of statistics. After computing the regression coefficient, they define the MCFS score for the $j$-th feature as follows.

$$MCFS(j) = \max_i |a_{i,j}|$$  \hspace{1cm} (5)

Similar to MCFS, MRSF uses another kind of sparse constraint. It aims to optimize the following problem.

$$\arg \min_{W} \| W^T X - Y \|_2$$  \hspace{1cm} (6)

$$s.t. \| W \|_{2,1} \leq \epsilon$$

where $W$ is the transformation matrix. $\epsilon$ is a predefined value. $\| W \|_{2,1}$ is the $\ell_{2,1}$ norm of $W$. (We will show the definition later.) Finally, MRSF use the $\ell_2$ norm of each row vector of $W$ to rank each feature. See more details in [30].

Intuitively, MCFS and MRSF employ different sparse constraints, i.e., $\ell_1$ and $\ell_{2,1}$, in constructing transformation matrix and use it in selecting features. Nevertheless, in these methods, the low dimensional embedding, i.e., $Y$, is determined in the first step and fixed in the following ranking or regression step. In other words, we do not consider the requirements of feature selection in deriving $Y$. If it can not only characterizes the manifold structure, but also indicate the requirements of regression, these methods would perform better.

B. Spectral Regression

Spectral Regression (SR) is a prominent framework for feature learning. In the literature, a lot of feature learning methods can be attributed to solving the following eigen-decomposition problem.

$$XAX^T w = \lambda XBX^T w,$$  \hspace{1cm} (7)

where $X$ is the data matrix, $A$ and $B$ are two symmetric matrices and $w$ is the computed projection vector.

SR aims to solve the above optimization problem in a two-stage way quickly. More concretely, it first computes the following eigen-problem to get $y$ and then computes the projection vector $w$ that can best fit the following equation in the least square sense.

$$Ay = \lambda B y,$$  \hspace{1cm} (8)

where $y$ can be regarded as the embedding of original data $x$. $y_i$ is the $i$-th component of $y$.

In essence, SR replaces $w$ by $y = w^T X$ to get the following function in Eq. (8). Through this kind of replacement, since the most important information is reserved in $y$, the performance is guaranteed and the computational complexity is reduced.

Considering the procedure of SR, we know that it can characterize the manifold structure. In other words, SR uses $y$ to replace the original data and compute a linear transformation matrix for the projection. Nevertheless, it aims to learn a subspace. We can not use it for feature selection directly.

III. Feature Selection via Joint Embedding Learning and Sparse Regression

In this section, we will first introduce some notations. The concrete formulation of our framework is then proposed. Finally, we take a special case within our framework for illustration and provide an effective algorithm to solve this problem.

Before going into the details of our algorithm, let us introduce some notations. Denote $\{x_i \in \mathbb{R}^d | i = 1, 2, \cdots, n\}$ as the unlabeled examples. We would like to select $s$ features to represent the original data, where $s < d$. For a matrix $Q \in \mathbb{R}^{u \times v}$, its $\ell_{r,p}$-norm is defined as follows.

$$\|Q\|_{r,p} = \left( \sum_{i=1}^{u} \left( \sum_{j=1}^{v} |Q_{ij}|^r \right)^{p/r} \right)^{1/p}.$$  \hspace{1cm} (9)

As seen from the above definition, when $p = r = 2$, it is the commonly used Frobenius norm or $\ell_2$ norm. In brief, we denoted it as $\| \cdot \|_2$ in the following. Note that, it is not equivalent to the 2-norm of the matrix. Define $\alpha > 0$, $\beta > 0$ as two balance parameters. In summary, we list the notations in Table I.

A. The Proposed framework

Considering the above analysis of traditional unsupervised feature selection methods, especially MCFS and MRSF, we
propose to use three objective functions in formulating our framework.

Let us introduce the first objective function. Considering that Spectral Regression (SR) performs well in feature learning and graph laplacian could fully characterize the manifold structure, we would like to inherit their advantages in formulating our feature selection algorithm. Evoked by the intuition that nearby points should have similar properties, we construct a weight graph \( G = (V, E, S) \) to reveal their local connections, where \( V = \{x_i\} \) is the graph vertex set and \( E \) contains edges of the constructed graph. \( S \) is the similarity matrix defined on this graph. The key point in constructing graph is to determine its weight matrix \( S \), where \( S_{ij} \) reveals the similarity between points \( x_i \) and \( x_j \).

Recalling the basic idea of feature learning, we will represent the original data \( x_i \) by its low dimensional embedding, i.e., \( y_i \in \mathbb{R}^m \), where \( m \) is the dimensionality of embedding. As a result, the first objective is

\[
\arg\min_{Y \in \mathbb{R}^{d \times n}} \sum_{i=1}^{n} \left\| y_i - \sum_{j=1}^{n} S_{ij} y_j \right\|^2 = tr(YLY^T) \tag{10}
\]

where \( L = (I_n \times n - S)^T(I_n \times n - S) \) is the graph laplacian, \( y_i \in \mathbb{R}^m \) is the embedding of \( x_i \) for \( i = 1, 2, \ldots, n \) and \( Y = [y_1, y_2, \ldots, y_n] \).

We now explain why we choose this objective function. In fact, we have represented the original data by its low dimensional embedding, i.e., \( y_i \). Through this kind of replacement, the most valuable information is retained and the feature redundancies are eliminated. It is commonly used in subspace learning.

In the above deduction, the most important step is to determine \( S \). Commonly, the graph is constructed by connecting every point to its \( k \)-nearest neighbors. Moreover, we construct a graph with the following step.

Step 1. Constructing a \( k \)-nearest graph \( G \). The \( i \)-th node corresponds to \( x_i \). For \( x_i \), it only connects with the points in its \( k \)-nearest neighborhood set \( N(x_i) \).

In the literature, there are many ways in determining the weights for connecting points. For example, we can use a Gaussian function like in Laplacian Eigenmaps[34] or local linear approximation weights like in Locally Linear Embedding (LLE)[36]. We can also employ the same strategies as in Locally tangent Space Alignment (LTSA)[37] or Local Spline Embedding (LSE) [38]. Among these methods, two are widely used. The first one is Gaussian function and the second way is local linear approximation weights. More concretely,

Step 2(a). Computing the similarity matrix \( S \). For the \( i \)-th point \( x_i \), its weight \( S_{ij} > 0 \) if and only if \( x_j \in N(x_i) \) or \( x_i \in N(x_j) \). Otherwise, \( S_{ij} = 0 \). The nonzero weight is determined by using the following Gaussian function.

\[
S_{ij} = \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma^2}\right) \tag{11}
\]

where \( \sigma \) is the width parameter.

Step 2(b). Computing the similarity matrix \( S \). For the \( i \)-th point \( x_i \), its weight \( S_{ij} > 0 \) if and only if \( x_j \in N(x_i) \). Otherwise, \( S_{ij} = 0 \). The nonzero weight is determined by using the following locally linear approximation strategy.

\[
\arg\min_{S} \sum_{i=1}^{n} \left\| x_i - \sum_{j \in N(x_i)} S_{ij} x_j \right\|^2. \tag{12}
\]

We now introduce the second optimization function. As in SR, the second objective function of our algorithm is to regress each sample to its low dimensional embedding. More concretely, assume \( \{x_i\} \) is centered and denote \( W = [w_1, w_2, \ldots, w_m] \in \mathbb{R}^{d \times m} \) as the matrix formed by all transformation vectors \( \{w_i\}_{i=1}^{m} \), where \( m \) is the dimensionality of \( y_i \). We define the second objective function as

\[
\arg\min_{W} \sum_{i=1}^{n} \left\| W^T x_i - y_i \right\|^2 = \left\| W^T X - Y \right\|^2. \tag{13}
\]

By optimizing this objective function, we compute the transformation matrix \( W \) which can minimize the regression error. The \( i \)-th row of transformation matrix \( W \) could be used to measure the importance of \( i \)-th feature of original data in the above regression. As shown later, by adding the sparse constraints, we can use the computed \( W \) to select features effectively.

The third objective function is designed for feature selection. As mentioned above, we denote \( \tilde{w}_i \) as the \( i \)-th row vector of \( W \), i.e.,

\[
W = [\tilde{w}_1^T, \tilde{w}_2^T, \ldots, \tilde{w}_d^T]^T. \tag{14}
\]

Essentially, \( \tilde{w}_i \) corresponds to the transformation vector of the \( i \)-th feature in regression. It can also be regarded as a vector that measures the importance of the \( i \)-th feature.

Considering the task of feature selection, we expect that the transformation matrix holds the sparsity property for feature selection. More concretely, we expect that only a few numbers of \( \tilde{w}_i \) are non-zeros. As a result, the corresponding features are selected since these features are enough to regress the original data \( x_i \) to its low dimensional representation \( y_i \). When we employ the \( r \)-norm of \( \tilde{w}_i \) as a metric to measure its contribution in this regression, the sparsity property, i.e., a small number of \( \tilde{w}_i \) are non-zeros, indicates the following objective function.

\[
\min_{W} \sum_{i=1}^{d} (\|\tilde{w}_i\|_r)^p = \sum_{i=1}^{d} \left( \sum_{j=1}^{m} |W_{ij}| \right)^{p/r} = \|W\|_{r,p}. \tag{15}
\]
Here \( \| W \|_{r,p} \) denotes the \( \ell_{r,p} \)-norm as defined in Eq. (9). For the sake of adding sparse properties to \( W \), we constrain that \( 0 \leq p \leq 2 \) and \( r \geq 2 \). This indicates that a small number of \( w_i \) are non-zeros. Intrinsically, different selection of \((r, p)\) means different kinds of approaches. We will summarize them later.

By combining the objective functions in Eq. (10), Eq. (13) with Eq. (15), our JELSR framework can be formulated as follows.

\[
\mathcal{L}(W, Y) = \arg \min_{W \in \mathbb{R}^{d \times m}} \min_{Y \in \mathbb{R}^{d \times m}} \quad \frac{1}{2} \| Y Y^T - I_{d \times m} \|_F^2 + \alpha \| W \|_{r,p}^2 + \beta \| W \|_1;
\]

where \( \alpha \) and \( \beta \) are two balance parameters.

After deriving \( W \), we use the \( r \)-norm of \( \hat{w}_i \), i.e., \( \| \hat{w}_i \|_r \) to rank the features. The larger \( \| \hat{w}_i \|_r \) is, the more important this feature is. We can either select a fixed number of the most important features or set a threshold and select the feature whose \( \| \hat{w}_i \|_r \) is larger than this value. In the following, we select a fixed number, i.e., \( s \), features for evaluation.

Intuitively, as seen from the formulation of JELSR in Eq. (16), we join two separate researches, i.e., embedding learning and sparse regression, for feature selection. As we will explain later, we can integrate the merits of feature learning and sparse regression. Thus, JELSR performs better.

Finally, we show that two prominent approaches, i.e., MCFS and MRSF, can be regarded as special cases of our framework. As mentioned above, the basic idea of MCFS is to first compute the low dimensional embedding \( Y \) and then regress each sample to its low dimensional embedding by adding \( \ell_1 \) norm regularization. Concretely, MCFS can be regarded as solving the following problems in sequence.

\[
\arg \min_{W \in \mathbb{R}^{d \times m}} 1 \quad \frac{1}{2} \| W^T X - Y \|_2^2 + \alpha \| W \|_{1,1}.
\]

Comparing the formulation in Eq. (17) with that in Eq. (16), we can see the main difference between JELSR and MCFS is that JELSR unifies the two objectives of MCFS. In other words, JELSR could join the procedures of embedding learning and sparse regression. MCFS separates these two steps. Thus, its performance is largely determined by the effectiveness of graph construction. More importantly, we have the following result.

**Property 1.** If we compute the graph laplacian \( L \) in Eq. (10) by employing Gaussian function in Eq. (11), MCFS in Eq. (17) can be regarded as a special case of JELSR in Eq. (16) when \( r = p = 1 \) and \( \beta \to 0 \).

**Proof:** When \( \beta \to 0 \), the Eq. (16) reduces to the first optimization problem in Eq. (17), namely, we need to solve \( Y \) at first. Then, the objective function in Eq. (16) is converted to the second problem in Eq. (17) to solve \( W \) when \( r = p = 1 \). In other words, when \( r = p = 1 \) and \( \beta \to 0 \), the solution of JELSR can be regarded as solving two problems in Eq. (17) in sequence, which is the formulation of MCFS.

Similarly, another state-of-art feature selection approach, i.e., MRSF, can also be analyzed within this framework. MRSF first computes the embedding by eigen-decomposition of graph laplacian and then regresses with \( \ell_{2,1} \) norm regularization. In other words, MRSF can be regarded as solving the following two problems in sequence.

\[
\arg \min_{W \in \mathbb{R}^{d \times m}} 1 \quad \frac{1}{2} \| W^T X - Y \|_2^2 + \alpha \| W \|_{2,1};
\]

As seen from the formulations in Eq. (16) and Eq. (18), we have the following results.

**Property 2.** If we compute the graph laplacian \( L \) in Eq. (10) by employing Gaussian function in Eq. (11), MRSF in Eq. (18) can be regarded as a special case of JELSR in Eq. (16) when \( r = 2 \), \( p = 1 \) and \( \beta \to 0 \).

The proof is the same as the Property 1. We would like to omit it.

One point should be mentioned here is that, although SR is not a feature selection approach, our framework also has a close relationship with SR, a famous dimensionality reduction approach. In fact, SR can be regarded as solving the following problem in a two stage way.

\[
\arg \min_{W \in \mathbb{R}^{d \times m}} 1 \quad \frac{1}{2} \| W^T X - Y \|_2^2 + \alpha \| W \|_{2,2}^2.
\]

Consequently, SR can be viewed as a special case within our framework when \( r = 2 \), \( p = 2 \) and \( \beta \to 0 \).

**Property 3.** If we compute the graph laplacian \( L \) in Eq. (10) by \( L = B^{-1} A \), SR in Eq. (19) can be regarded as a special case of JELSR in Eq. (16) when \( r = 2 \), \( p = 2 \) and \( \beta \to 0 \).

In summary, the above analyses indicate that JELSR can be regarded as a unified framework in viewing different learning based feature selection approaches and SR. Traditional feature selection methods separate the procedures of subspace learning and sparse regression. Different approaches use different kinds of sparse regularization. The relationships are listed in Fig. 1.

**B. JELSR Formulations and Solution**

In this section, to show the effectiveness of the proposed framework, we will use local linear approximation weights and add the \( \ell_{2,1} \) norm regularization. In other words, we use Eq. (12) in Step 2(b) to compute \( S \) and employ \( \ell_{2,1} \) norm regularization in regression. The reason why we take it as
an example is as follows. (1) In many applications, we have prominent performance in using locally linear approximation weight to construct graph[39]. (2) As defined in Eq. (9), p is set to add sparse constraint for feature selection. Thus, 0 ≤ p ≤ 1. If we choose p = 0, the formulated problem is not convex and it is hard to solve. We assume p = 1. Besides, since r is set to measure the norm of each row vector, it is often assumed that r ≥ 2. The reason why we choose r = 2 is also for solution. When r > 2, it will not take large influence on the final results. In a word, the ℓ_{2,1} norm regularization satisfied our requirement for feature selection and indicated that only a small number of \( w_i \) are non-zeros.

More concretely, the formulation of JELSR is

\[
\mathcal{L}(W, Y) = \arg \min_{W, Y: YY^T = I_{m \times m}} \text{tr}(YLY^T) + \beta(\|W^TX - Y\|_2^2 + \alpha\|W\|_{2,1}).
\] (20)

Considering the optimization problem in Eq. (20), since we have added the ℓ_{2,1}-norm regularization for feature selection, it is hard to derive its closed solution directly. Inspired by [40], we will solve this problem in an alternative way. As we will explain later, through this kind of procedure, we convert the problem with a couple of variables (W and Y) into one variable (Y) and then update the sparse regression matrix W. In other words, we select the features by joining embedding learning and sparse regression, which has not been considered in the literature.

Note that \( \|W\|_{2,1} \) is convex. Nevertheless, its derivative does not exist when \( \bar{w}_i = 0 \) for \( i = 1, 2, \cdots, d \). For convenience, we would like to denote \( \mathcal{L}(W) = \|W^TX - Y\|_2^2 + \alpha\|W\|_{2,1} \). Thus, when \( \bar{w}_i \neq 0 \) for \( i = 1, 2, \cdots, d \), the derivative of \( \mathcal{L}(W) \) with respect to W is

\[
\frac{\partial \mathcal{L}(W)}{\partial W} = 2XX^TW - 2XY^T + 2\alpha UW,
\] (21)

where \( U \in \mathbb{R}^{d \times d} \) is a diagonal matrix whose \( i \)-th diagonal element is

\[
U_{ii} = \frac{1}{2\|\bar{w}_i\|_2^2}.
\] (22)

As seen from Eq. (21), we construct an auxiliary function. When U is fixed, the derivative in Eq. (20) can also be regarded as the derivative of the following objective function.

\[
\mathcal{L}(W) = \|W^TX - Y\|_2^2 + \alpha\text{tr}(W^TUW).
\] (23)

Consequently, we try to solve the following problem to approximate the solution to Eq. (20).

\[
\mathcal{L}(W, U, Y) = \arg \min_{W: YY^T = I_{m \times m}} \text{tr}(YLY^T) + \beta(\|W^TX - Y\|_2^2 + \alpha\text{tr}(W^TUW))
\] (24)

where \( U \) is defined as in Eq. (22).

We would like to explain why we can derive a sparse solution by minimizing Eq. (24). Recalling the definition of \( U_{ii} \) in Eq (22), we know that \( \text{tr}(W^TUW) = \|W\|_{2,1}/2 \) when \( \bar{w}_i \) is not equal to 0. Thus, the objective of minimizing \( \text{tr}(W^TUW) \) will add the sparsity constraint on W. Intuitively, if \( \|\bar{w}_i\|_2 \) is small, then \( U_{ii} \) is large and the minimization of Eq. (23) tends to derive \( w_i \) with much smaller ℓ_2-norm. After several times of iteration, the norms of some \( w_i \)s are close to zero and we get a sparse W. Besides, if \( \bar{w}_i = 0 \), we will add a regularizer \( \|w_i\|_2 + \epsilon \) to replace \( \|\bar{w}_i\|_2 \) in our implementation, where \( \epsilon > 0 \) is a very small value.

As seen from above formulation, the objective function in Eq. (24) is convex with respect to W and Y if U is fixed. When W is fixed, we can determine U by Eq. (22) directly. Thus, we update W and Y when U is fixed and compute U when W is fixed.

When U is fixed, we would like to take the derivative of \( \mathcal{L}(W, U, Y) \) with respect to W and set it to zero, i.e., we have the following equation.

\[
\frac{\partial \mathcal{L}(W, U, Y)}{\partial W} = 2XX^TW - 2XY^T + 2\alpha UW = 0
\] (25)

or equivalently,

\[
W = (XX^T + \alpha U)^{-1}XY^T.
\] (26)

By substituting above W into Eq. (24), we will have

\[
\mathcal{L}(W, U, Y) = \text{tr}(YLY^T) + \beta(\|W^TX - Y\|_2^2 + \alpha\text{tr}(W^TUW))
\]

\[
= \text{tr}(YLY^T) + \beta(\text{tr}(W^TXX^TW) - 2\text{tr}(W^TYX^T) + \text{tr}(YY^T) + \alpha\text{tr}(W^TUW))
\]

\[
= \text{tr}(YLY^T) + \beta(\text{tr}(W^TXX^TW + \alpha U)W) + \text{tr}(YY^T)
\]

(27)

Denote A = XX^T + \alpha U, Eq. (27) becomes

\[
\mathcal{L}(W, U, Y) = \text{tr}(Y(L + \beta I_{n \times n} - \beta X^T A^{-1} X) Y^T)
\]

(28)

Considering the objective function in Eq. (28) and the constraint YY^T = I_{m \times m}, the optimization problem becomes

\[
\arg \min_Y \text{tr}(Y(L + \beta I_{n \times n} - \beta X^T A^{-1} X) Y^T)
\]

s.t. YY^T = I_{m \times m}

(29)

If A and L are fixed, the optimization problem in Eq. (29) can be solved by eigen-decomposition of matrix \( (L + \beta I_{n \times n} - \beta X^T A^{-1} X) \). We pick up the eigenvectors corresponding to the m smallest eigenvalues.

When W is fixed, we can update U by employing the formulation in Eq. (26) directly.

In summary, we solve the optimization problem in Eq. (24) in an alternative way. More concretely, if U is fixed, we can first solve the optimization problem in Eq. (29) to update Y and then employ Eq. (26) to update W. After that, we fix W and update U, which is defined in Eq. (22). Note that, although we combine embedding learning and sparse regression in Eq. (20), our algorithm is not a simple alternation between them.

We now explain why our method could combine embedding learning and sparse regression. Considering the above algorithm, we solve the problem in Eq. (29) to compute Y. In other words, the objective of sparse regression has also affected the derivation of low-dimensional embedding, i.e., Y. Traditional methods, such as MCFS and MRSF, minimize tr(YLY^T) at one time and it is fixed in the following feature selection step.
Additionally, since JELSR is solved in an alternative way, we would like to initialize U by an identity matrix. The experimental results show that our algorithm converges fast by using this kind of initialization. The number of iterations is often less than twenty.

In summary, the procedure of JELSR is listed in Table II.

### IV. DISCUSSIONS

In this section, we will analyze JELSR in three aspects. We first provide the convergence behavior and then discuss computational complexity and parameter determination problems.

#### A. Convergence Analysis

Since we have solved JELSR in an alternative way, we would like to show its convergence behavior. First, a lemma [40] is provided.

**Lemma 1.** For any non-zero vectors \( \mathbf{a}, \mathbf{b} \in \mathbb{R}^m \), the following result follows

\[
\|\mathbf{a}\|_2 - \frac{\|\mathbf{a}\|_2^2}{2\|\mathbf{b}\|_2} \leq \|\mathbf{b}\|_2 - \frac{\|\mathbf{b}\|_2^2}{2\|\mathbf{b}\|_2}.
\]

(30)

The convergence behavior of JELSR is summarized in the following theorem.

**Proposition 1.** The objective of the problem in Eq. (20) in each iteration is non-increasing by employing the optimization procedure in the second stage of Table II.

**Proof:** As seen from the algorithm in Table II, when we fix U as \( U^t \) in the \( t \)-th iteration and compute \( W^{t+1}, Y^{t+1} \), the following holds,

\[
\mathcal{L}(W^{t+1}, U^t, Y^{t+1}) \leq \mathcal{L}(W^t, U^t, Y^t)
\]

(31)

Since \( \|W\|_{2,1} = \sum_{i=1}^{d} \|\hat{w}_i\|_2 \), the above inequality indicates

\[
\begin{align*}
&\text{tr}(Y^{t+1}L(Y^{t+1})^T) + \beta(\|W^{t+1}\|_F^2 X - Y^2) \\
&+ \alpha \|W^{t+1}\|_{2,1} + \alpha \sum_{i=1}^{d} \left( \frac{\|\hat{w}_i\|_2}{2\|\hat{w}_i\|_2^2} - \|\hat{w}_i\|_2 \right) \\
&\leq \text{tr}(Y^TL(Y^T)^T) + \beta(\|W^T\|_F^2 X - Y^2) \\
&+ \alpha \|W^T\|_{2,1} + \alpha \sum_{i=1}^{d} \left( \frac{\|\hat{w}_i\|_2}{2\|\hat{w}_i\|_2^2} - \|\hat{w}_i\|_2 \right).
\end{align*}
\]

(32)

Recalling the results in Lemma 1, we know that

\[
\frac{\|\hat{w}_i^{t+1}\|_2^2}{2\|\hat{w}_i^{t+1}\|_2} - \|\hat{w}_i^{t+1}\|_2 \geq \frac{\|\hat{w}_i^t\|_2^2}{2\|\hat{w}_i^t\|_2} - \|\hat{w}_i^t\|_2.
\]

(33)

Combining Eq. (32) with Eq. (33), we have the following results.

\[
\begin{align*}
\text{tr}(Y^{t+1}L(Y^{t+1})^T) + \beta(\|W^{t+1}\|_F^2 X - Y^2) + \alpha \|W^{t+1}\|_{2,1} \\
\leq \text{tr}(Y^TL(Y^T)^T) + \beta(\|W^T\|_F^2 X - Y^2) + \alpha \|W^T\|_{2,1}.
\end{align*}
\]

(34)

This inequality indicates that the objective function in Eq. (20) will monotonically decrease in each iteration.

Additionally, since the objective function has lower bounds, such as zero, the above iteration will converge. Besides, we have conducted some experiments to show whether the objective function is non-increasing. They are listed in Section V(C). One point should be highlighted here. The above theorem only indicates that the objective function is non-increasing. Nevertheless, we do not know whether W converges. Thus, we will also provide some results for illustration.

Since W is used for feature selection, we would like to measure the variance between two sequential Ws by the following metric.

\[
\text{Error}(t) = \sum_{i=1}^{d} \|\hat{w}_i^{t+1}\|_2 - \|\hat{w}_i^t\|_2
\]

(35)

It will guarantee that the final feature results will not be changed drastically. The following experiments also show that the proposed method converges fast. Number of iterations is less than 20.

#### B. Computational Complexity Analysis

Since all the methods use different metrics to rank features in the end, we only compare their computational complexities in evaluating each feature. Pcascor only needs to compute the variance of each dimensionality. Its computational complexity is \( O(dn) \). The most time consuming step of LapScor is the construction of laplacian matrix. The computational complexity of LapScor is \( O(n^2d) \). SPEC costs the most time in decomposing the laplacian matrix and its computational complexity is also \( O(n^3) \).

MCFS first decomposes a laplacian matrix and then solve a least square regression problem with \( \ell_1 \) norm regularization by Lasso. Their computational complexities are \( O(n^2) \) and \( O(mnd) \). Thus, its computational complexity is \( \max\{O(n^2), O(n^2d), O(mnd)\} \). As stated in [30], MRSF uses an iterative algorithm to solve the problem in Eq. (18). Its computational complexity is \( \max\{O(n^2d), O(s^2m + s^2d)\} \), where \( s \) is the number of selected features.

Compared with other related approaches, the computational complexity of our algorithm is not so high. In fact, the most computational step of JELSR is to solve the problem in Eq. (26) and Eq. (27) alternatively. Their computational complexities are \( O(mnd) \) and \( O(n^3) \). Thus, the computational complexity of JELSR is \( \max\{O(n^2d), O(n^3), O(mnd)\} \). We will provide some experimental results to compare the computational time in the next section.
Finally, one point should be explained here. In the above analyses, we only consider the computational complexity theoretically. In real applications, however, the time consumption may be different. This is due to the following reasons: (1) Different implementations of the same method may cost different time; (2) We have not considered the influence of iteration in the above analysis. Nevertheless, it takes influence in many real applications.

C. Parameter Determination

Parameter determination is still an open problem. The first parameter is \( m \). It is known as intrinsic dimensionality in the literature of manifold learning. One common way is to determine it by grid search. We determine it as in traditional approaches, such as [41]. The second parameter is the number of selected features. It is difficult to determine it without prior. Thus, we vary this parameter within a certain range and show its influence as in MCFS and MRSF. Finally, another two parameters, i.e., \( \alpha \) and \( \beta \) are empirically determined by grid search. We will also present some numerical results to show their influence.

V. Experiments

Our method will be evaluated in typical unsupervised and supervised tasks, i.e., clustering and classification. We present six different groups of experiments. The first experiment is to provide a toy example. The second group is to show some results about convergence behavior. The third group contains the clustering results of Kmeans on different data with different numbers of selected features. The fourth group consists of classification results. To compare the computational efficiency, we also provide some numerical results. Since there are mainly two different parameters, i.e., \( \alpha \) and \( \beta \), we would like to provide the results with different parameters. These parameters are determined by grid search in the following experiments.

A. Data Description and Evaluation Metric

There are mainly six different types of matrix data sets. They are images, including Umist\(^1\), Orl\(^2\) and Coil20\(^3\), voice data, including Isolet\(^4\), Sonar data \(^5\) and biological data, i.e., Lymph \(^6\). Their sizes range from about 100 to about 1500. The dimensionality ranges from about 60 to about 4000. The number of clusters ranges from 2 to 40. We select different numbers of features and these data sets are employed to validate our algorithms. The character of these data sets is listed in Table III.

To show the effectiveness of our method, we use two different kinds of tasks on data sets formulated by the selected features. One task is unsupervised, i.e., clustering and the other is supervised, i.e., classification. In the following experiments, a popular clustering methods, i.e., K-mean(Km), is employed to cluster data with selected features. We also use two different metrics to evaluate the performances of clustering. One is the Clustering Accuracy (ACC) defined as

\[
\text{ACC} = \frac{1}{n} \sum_{i=1}^{m} \delta(l_i, \text{map}(c_i)),
\]

where \( l_i \) is the actual label and \( c_i \) is the computed cluster index. \( \delta(\cdot) \) represents the \( \delta \)-function. \( \text{map}(\cdot) \) is a function that maps each cluster index to the best class label. It can be found by the Hungarian algorithm [42]. Another one is Normalized Mutual Information (NMI) [43]. For classification, we use the Nearest Neighborhood classifier (NN) and compute the classification accuracy. Some statical analyses are also provided.

B. Toy Example

We randomly select two samples from each class of the Orl data set as the training data. The rest is considered as testing examples. By using our method on the training data, we select \{128, 256, 384, 512, 640, 768, 896, 1024\} features. Then, we randomly select face images from two testing samples. For illustration, when the unselected features are set to white and the selected features maintain their original values, we can draw them in Fig.2. From left to right, the number of selected features are \{128, 256, 384, 512, 640, 768, 896, 1024\} respectively.

![Fig. 2. Toy example. Top: a test sample of Orl data from the first class with different number of selected features; Bottom: a test sample of Orl data from the ninth class with different number of selected features.](http://llmpp.nih.gov/lymphoma/data.shtml)

<table>
<thead>
<tr>
<th>Data</th>
<th>Size</th>
<th>Dim</th>
<th>Class</th>
<th>Type</th>
<th># of Selected Fea</th>
</tr>
</thead>
<tbody>
<tr>
<td>Umist</td>
<td>575</td>
<td>644</td>
<td>20</td>
<td>Image</td>
<td>20, 30, · · · , 110</td>
</tr>
<tr>
<td>Isolet</td>
<td>1559</td>
<td>617</td>
<td>26</td>
<td>Voice</td>
<td>20, 40, · · · , 200</td>
</tr>
<tr>
<td>Orl</td>
<td>400</td>
<td>1024</td>
<td>40</td>
<td>Image</td>
<td>20, 30, · · · , 110</td>
</tr>
<tr>
<td>Sonar</td>
<td>208</td>
<td>60</td>
<td>2</td>
<td>Voice</td>
<td>3, 5, · · · , 21</td>
</tr>
<tr>
<td>Coil20</td>
<td>1440</td>
<td>1024</td>
<td>20</td>
<td>Image</td>
<td>20, 30, · · · , 110</td>
</tr>
<tr>
<td>Lymph</td>
<td>96</td>
<td>4026</td>
<td>8</td>
<td>Biology</td>
<td>20, 40, · · · , 200</td>
</tr>
</tbody>
</table>

\(^1\)http://images.ear.umist.ac.uk/danny/database.html
\(^2\)http://www.zjuacdeg.cn/dengcai/Data/faceData.html
\(^3\)http://www1.cs.columbia.edu/CAVE/research/softlib/coil-20.html
\(^4\)http://archive.ics.uci.edu/ml/machine-learning-databases/isolet/
\(^5\)http://archive.ics.uci.edu/ml/machine-learning-databases/sonar/
\(^6\)http://llmpp.nih.gov/lymphoma/data.shtml

C. Convergence Results

In this section, we provide some numerical results to show the convergence behavior of our iterative procedure. Three data sets, i.e., Umist, Isolet and Coil, are employed. We set the parameter \( m = c \) and other parameters, such as \( k \), are empirically determined as in traditional approaches[36]. We employ all data as the input and provide two kinds of results.
The first is about the objective function and the other is the divergence between two consecutive $W$. The evaluation metric has been shown in Eq. (35). All the results with 20 iterations are shown in Fig. 3.

As seen from the Fig.3, the objective of JELSR is non-increasing during the iteration. They all converge to a fixed value. Additionally, the divergence between two consecutive $W$ converges to zero, which means that the final results will not be changed drastically. Besides, the convergence is very fast. Times of iteration is less than 10.

**D. Clustering Results With Kmeans**

We evaluate our method in a typical unsupervised task, i.e., clustering. We compare our method with other famous learning based feature selection approaches, including PcaScor, LapScor, SPEC, MCFS and MRSF. We also compare with Unsupervised Fuzzy Rough Set Feature Selection (UFRFS) approach by employing Weka. After employing these methods to select features on the total data set, we then use Kmeans to cluster the corresponding data formulated by the selected features. Since the performance of Kmeans is largely dominated by initialization [44], it is repeated for 100 independent times. With different numbers of selected features, the mean and standard derivation (std) values of ACC and NMI results are shown in Fig. 4, Table IV-V, where the parameters $\alpha$ and $\beta$ are selected by grid search in a heuristic way. Other parameters are empirically determined as in traditional subspace learning approaches, such as SR. Note that, in Table IV-VII, the boldfaced values are the highest in statistical view. More concretely, only if a number subtracts its std is larger than another number adds its std, we can call that the first number is larger. Otherwise, we think that the first one is not larger than another significantly in statistical view. Besides, since UFRFS only searches the optimal feature set that can satisfy predefined measurement, the number of selected features is often within a small range. Thus, we only report its results on this small range within a subfigure.

As seen from Fig. 4, Table IV-V, we have the following conclusions. (1) When we use the clustering accuracy to measure the performance of different feature selection approaches, JELSR performs better than other approaches in most cases, no matter what the number of selected feature is. (2) Although ACC and NMI are two different metrics, they both indicate the advantages of our algorithm. (3) With the increase of $s$, the clustering results do not always increase. It indicates that a large number of selected features does not always helpful for clustering. It may be caused by the addition of redundancy when we select more features.

**E. Classification with NN**

In this section, we compare our method with other approaches in supervised scenario. As in [30], we randomly sample 50% samples as the training data and the remaining are used for the test. The process is repeated for 100 times and results in 100 different partitions. First, we use the training data as the input of feature selection approaches and determine the selected features. Then, NN is employed for classification, where previous training data with selected features are determined as training samples and original unlabeled data with selected features are testing examples. Besides, we also compare one of the famous supervised feature selection approach ReliefF in this scenario [45]. With different number of selected features, we have conducted experiments on three data sets, i.e., Umist, Isolet and Sonar. As in [30], other parameters are turned by cross validation if necessary. The
mean classification accuracy is shown in Fig. 5.

Besides, we compare our method with other approaches by Student’s t-test. The statistical significance with a threshold of 0.05 of listed in Table VI. In this table, the first letter means the comparing results and the value in brackets in the corresponding p-value. "W" means that JELSR performs better than other approaches and "F" means that JELSR fails and the corresponding p-value is the probability that the other method performs worse than JELSR. "B" means that we can not distinguish them in statistical view. In this case, we have not reported the p-value when the mark is "B".

### F. Computational Complexity Comparison

The fourth group of experiments is proposed to compare the computational complexity. To show the influence of data size n and dimensionality d, we select two representative data sets, i.e., Isolet and Lymph, since they have the largest n and d among six data sets. We select s = 10, s = 30 and s = 50 features of Isolet and s = 20, s = 100 and s = 200 features for Lymph. With a naive MATLAB implementation, the calculations are made on a 3.2-GHz Windows machine. The computational time with grid search in determining the calculations are made on a 3.2-GHz Windows machine.

#### TABLE VI

<table>
<thead>
<tr>
<th>Number of Features</th>
<th>PcaScor</th>
<th>LapScor</th>
<th>SPEC</th>
<th>MCFS</th>
<th>MRSF</th>
<th>JELSR</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.0147</td>
<td>0.0169</td>
<td>0.0176</td>
<td>0.0146</td>
<td>0.0151</td>
<td>0.0157</td>
</tr>
<tr>
<td>10</td>
<td>0.0121</td>
<td>0.0138</td>
<td>0.0146</td>
<td>0.0134</td>
<td>0.0145</td>
<td>0.0155</td>
</tr>
<tr>
<td>15</td>
<td>0.0107</td>
<td>0.0124</td>
<td>0.0134</td>
<td>0.0125</td>
<td>0.0139</td>
<td>0.0145</td>
</tr>
<tr>
<td>20</td>
<td>0.0093</td>
<td>0.0111</td>
<td>0.0126</td>
<td>0.0123</td>
<td>0.0134</td>
<td>0.0140</td>
</tr>
<tr>
<td>25</td>
<td>0.0082</td>
<td>0.0101</td>
<td>0.0117</td>
<td>0.0120</td>
<td>0.0130</td>
<td>0.0135</td>
</tr>
<tr>
<td>30</td>
<td>0.0072</td>
<td>0.0093</td>
<td>0.0107</td>
<td>0.0114</td>
<td>0.0126</td>
<td>0.0130</td>
</tr>
<tr>
<td>35</td>
<td>0.0062</td>
<td>0.0085</td>
<td>0.0101</td>
<td>0.0110</td>
<td>0.0122</td>
<td>0.0125</td>
</tr>
<tr>
<td>40</td>
<td>0.0055</td>
<td>0.0077</td>
<td>0.0097</td>
<td>0.0105</td>
<td>0.0118</td>
<td>0.0120</td>
</tr>
<tr>
<td>45</td>
<td>0.0047</td>
<td>0.0069</td>
<td>0.0094</td>
<td>0.0102</td>
<td>0.0114</td>
<td>0.0117</td>
</tr>
<tr>
<td>50</td>
<td>0.0041</td>
<td>0.0061</td>
<td>0.0087</td>
<td>0.0095</td>
<td>0.0108</td>
<td>0.0110</td>
</tr>
</tbody>
</table>

As seen from the results in Table VII and VIII, we can draw the following conclusions. (1) PcaScor costs the least time in...
TABLE VIII

<table>
<thead>
<tr>
<th>s</th>
<th>PcaScor</th>
<th>LapScor</th>
<th>SPEC</th>
<th>MCFS</th>
<th>MRSF</th>
<th>JELSR</th>
</tr>
</thead>
<tbody>
<tr>
<td>s=20</td>
<td>0.5113</td>
<td>0.0469</td>
<td>0.4844</td>
<td>20.143</td>
<td>759.78</td>
<td>7322.15</td>
</tr>
<tr>
<td>s=100</td>
<td>0.5113</td>
<td>0.0781</td>
<td>0.5156</td>
<td>372.781</td>
<td>780.18</td>
<td>7534.67</td>
</tr>
<tr>
<td>s=200</td>
<td>0.5113</td>
<td>0.0625</td>
<td>0.5156</td>
<td>716.349</td>
<td>759.78</td>
<td>7322.15</td>
</tr>
</tbody>
</table>

Fig. 6. ACC and NMI of Kmeans on Umist and Isolet data sets with different $\alpha$ and $\beta$. (a) The ACC results on Umist. $\alpha$ varies from 1.5 to 2.4 and $\beta$ varies from 1e-2 to 1e-1. (b) The NMI results on Umist. $\alpha$ varies from 1.5 to 2.4 and $\beta$ varies from 1e-2 to 1e-1. (c) The ACC results on Isolet. $\alpha$ varies from 0.5 to 1.4 and $\beta$ varies from 0.1 to 1. (d) The NMI results on Isolet. $\alpha$ varies from 0.5 to 1.4 and $\beta$ varies from 0.1 to 1.

VI. CONCLUSION

In this paper, we aim to provide insights into the relationship among the state-of-the-art unsupervised feature selection approaches, as well as to facilitate the design of new algorithms. A general framework named as JELSR has been proposed to provide a unified perspective for the understanding and comparison of many popular unsupervised feature selection algorithms. Moreover, this framework can be used as a general platform to develop new algorithms. As shown in this paper, we have proposed a novel unsupervised feature selection algorithm that is shown to be more effective. A byproduct of this paper is a series of theoretical analysis and some interesting optimization strategies. One of our future works is to systematically compare all possible extensions of the algorithms developed by different configurations of $r$ and $p$, including its theoretical analyses and solving strategies. Another open problem is the selection of parameter $\alpha$ and $\beta$, which is an unsolved problem in many learning algorithms. In this paper, they are empirically determined. Additional theoretical analysis is also needed for this topic.

REFERENCES


