Feature Selection by Joint Graph Sparse Coding

Xiaofeng Zhu*and Xindong Wu†and Wei Ding‡and Shichao Zhang§

Abstract

This paper takes manifold learning and regression simultaneously into account to perform unsupervised spectral feature selection. We first extract the bases of the data, and then represent the data sparsely using the extracted bases by proposing a novel joint graph sparse coding model, JGSC for short. We design a new algorithm TOSC to compute the resulting objective function of JGSC, and then theoretically prove that the proposed objective function converges to its global optimum via the proposed TOSC algorithm. We repeat the extraction and the TOSC calculation until the value of the objective function of JGSC satisfies pre-defined conditions. Eventually the derived new representation of the data may only have a few non-zero rows, and we delete the zero rows (a.k.a. zero-valued features) to conduct feature selection on the new representation of the data. Our empirical studies demonstrate that the proposed method outperforms several state-of-the-art algorithms on real datasets in term of the *kNN* classification performance.

1 Introduction

Feature selection is an effective solution to the high-dimensionality problem in real applications. Recent studies on spectral feature selection have integrated manifold learning into features selection. Spectral feature selection can improve the performance of feature selection because it preserves the local structures of the data via manifold learning. However, most existing efforts are designed to sequentially conduct manifold learning and regression. For example, the MCFS method in [2] (see Fig. 1.(a)) includes two key steps, manifold learning and spectral regression respectively. After performing manifold learning, i.e., performing locality preserving projection (LPP), MCFS conducts spectral regression one eigenvector at a time to generate the element sparsity (see Fig. 2.(a)). Then a new score rule is designed to rank the goodness of the features (with the element sparsity). At last MCFS performs feature selection by deleting the features with low score values.

*The University of Queensland, Australia
†The University of Vermont, USA
‡The University of Massachusetts Boston, USA
§Guangxi Normal University, China

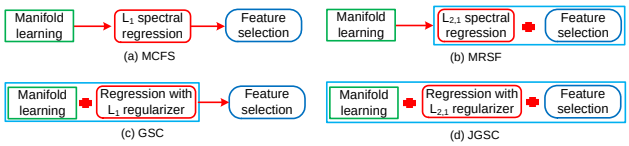
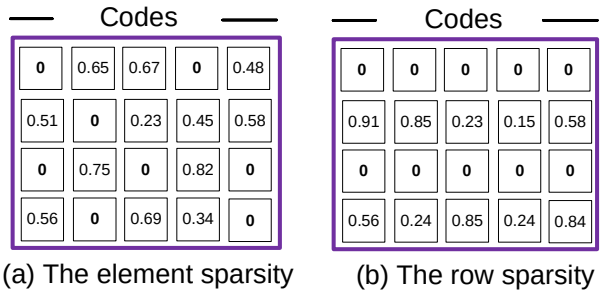


Figure 1: Methods on spectral feature selection.

MCFS has the following limitations. Firstly, the element sparsity is ill-fitted for feature selection. Given an example shown in Fig. 2.(a), according to the score rule in MCFS, the second row will be the first feature to be deleted for feature selection. However, the non-zero values in the second row (i.e., 0.51, 0.23, 0.45 and 0.58) will be lost. It is obvious that no matter which row is deleted, the non-zero values in that row will be lost since the element sparsity does not generate zero elements through the whole row, which is different from the row sparsity illustrated in Fig. 2.(b). Secondly, MCFS conducts spectral regression one eigenvector at a time, thus it does not consider the correlations among the spectral features. Recent studies (e.g., [15]) have shown that evaluating the spectral features individually cannot effectively identify redundant features. Thirdly, the performance of MCFS can be further improved by simultaneously performing manifold learning and regression, such as the GSC and JGSC methods (see Fig. 1.(c) and Fig. 1.(d)). Existing literatures (e.g., [3, 16]) have shown that simultaneously considering manifold learning and regression results improved performance in learning, e.g., classification or clustering.

Zhao et al., [15] proposed the MRSF algorithm (see Fig. 1.(b)) to perform spectral regression after conducting manifold learning. MRSF uses the $\ell_{2,1}$-norm regularizer to replace the ℓ_1-norm regularizer in MCFS. As illustrated in Fig. 2, the row sparsity used in MRSF obviously fits better for feature selection than the element sparsity used in MCFS. However, MRSF only solves the first two problems of MCFS but does not address the third one.

Recently graph sparse coding (GSC) [3, 16] (see Fig. 1.(c)) embeds manifold learning into the reconstruction process. It has been shown that GSC achieves improved performance in real applications. Due to employing



(a) The element sparsity (b) The row sparsity

Figure 2: An illustration to compare the element sparsity (via the ℓ_1-norm regularizer) with the row sparsity (via the $\ell_{2,1}$-norm regularizer).

the ℓ_1-norm regularizer for achieving the sparsity, GSC generates the element sparsity (see Fig. 2.(a)). GSC only focuses on the third problem of MCFS and does not address the first two issues. Moreover, GSC is designed for clustering [16] and classification [3] rather than feature selection.

In this paper we discuss a new feature selection method to overcome the drawbacks of MCFS. Motivated by existing studies (e.g., [2, 3, 15, 16]), we model feature selection as a novel joint graph sparse coding model (JGSC). The proposed JGSC (see Fig. 1.(d)) is designed to simultaneously perform manifold learning and regression as well as to achieve the row sparsity. In particular, JGSC consists of three key steps: 1) *Basis extraction.* The bases are derived from the data via an existing dictionary learning method, such as [8]. 2) *Data reconstruction.* The data is reconstructed into the derived basis space to generate its new representation using the proposed new joint graph sparse coding model. We repeat these two steps (i.e., basis extraction and data reconstruction respectively) until the value of the objective function of JGSC satisfies pre-defined conditions. 3) *Feature selection.* Due to introducing the $\ell_{2,1}$-norm regularizer, the derived representation of the data contains many zero rows. This shows that the zero rows (a.k.a. zero-valued features) of the new representation of the data are unimportant. To achieve efficiency and effectiveness, we delete those rows to obtain a reduced dataset.

We summarize the contributions of this paper as follows:

- We identify limitations existing in traditional spectral feature selection, mainly caused by unavoidable drawbacks of MCFS discussed above. We devise an effective solution to tackle the limitations via the proposed joint graph sparse coding model by simultaneously performing manifold learning and regression (between the original data and the bases, also called as reconstruction) with the $\ell_{2,1}$-norm regularizer.
- The proposed JGSC employs the least square loss function to achieve the minimal reconstruction error, and uses the graph Laplacian regularizer (i.e., manifold learning) to preserve the local structures of the data and consider the correlations among the data. JGSC also employs the $\ell_{2,1}$-norm regularizer to achieve the goals, such as avoiding the issue of over-fitting, achieving the row sparsity and considering the correlations among the features.
- The proposed model performs feature selection on the new representation of the data (or in the basis space), rather than performing feature selection on the original data (or in the original space) used in traditional feature selection methods, such as [2, 15, 13]). Using the bases to represent the data has been proven to be a *higher*-level and more *abstract* representation than the traditional ones, such as raw pixel intensity values. Compared to the traditional methods, the high-level representation makes the learning process easier and leads to better results in practice [8, 11]. Furthermore, extensive experimental results on the benchmark datasets show that the proposed model is more effective than state-of-the-art methods.

The remainder parts are organized as below: Section 2 reviews related work. Section 3 gives the details of the proposed approach followed by its theoretical analysis in Section 4. The experimental results are reported and analyzed in Section 5 while Section 6 concludes the paper.

2 Related work

In this section, we give a brief review of feature selection and sparse learning, and describe the notations used in this paper.

2.1 Feature selection Given a data set with a large number of features, if some of them are irrelevant, feature selection performs dimension reduction by removing the irrelevant features, and then outputting the relevant features. Feature selection is popular in many real-world applications [17], such as information retrieval, image analysis, intrusion detection, bioinformatics, and so on.

During the process of feature selection, the training data can be either labeled, or unlabeled, or partially labeled. This leads to supervised feature selection (e.g., [10]), unsupervised feature selection (e.g., the method

in [13], MCFS, MRSF and the proposed JGSC) and semi-supervised feature selection (e.g., [12]).

According to design strategies, existing feature selection methods can be broadly categorized into three groups: the filter approach, the wrapper approach and the embedded approach [6]. In real applications, the filter approach (e.g., [4]) is robust against the issue of over-fitting, but may fail to select the most "useful" features. The wrapper approach (e.g., [9]) can in principle find the most "useful" features, so often outperforms the filter approach. However, the wrapper approach needs high computational cost and is prone to the issue of over-fitting. The embedded approach (e.g., the method in [13], MCFS, MRSF and our proposed JGSC) can obtain superior performance and efficiency than the other two. Comparing with the filter approach, both wrapper and embedded approaches can obtain higher performance because they always select an optimal feature subset. Comparing with the wrapper approach, the embedded approach needs less computational cost and is less prone to the issue of over-fitting.

2.2 Sparse learning Sparse learning distinguishes important elements from unimportant ones by assigning the codes of unimportant elements as zero and the important ones as non-zero. This enables that sparse learning reduces the impact of noises and increase the efficiency of learning models [8].

According to the way to generate sparsity patterns, we categorize existing sparse learning into two categories, separable sparse learning (e.g., lasso [8], group lasso [14], MCFS and GSC, see Fig. 2.(a)), and joint sparse learning (e.g., MRSF, and the proposed JGSC, see Fig. 2.(b)). Separable sparse learning encodes one sample individually. Joint sparse learning simultaneously encodes all samples once. For example, the codes of five samples are shown in each subfigure of Fig. 2. Each column is the codes of one sample, where a zero element means sparse code and non-zero element means dense code (or non-sparse code). In this example, to generate codes for all five samples, separable sparse learning needs to perform its objective function five times, while joint sparse learning only performs once for all samples.

Sparse learning employs different regularizers to lead to different sparse patterns. For example, the ℓ_1-norm regularizer (e.g., [8]) leads to the element sparsity (Fig. 2.(a)) on separable sparse learning; the $\ell_{2,1}$-norm regularizer leads to the row sparsity Fig. 2.(b) by considering the correlation among the training data on joint sparse learning.

2.3 Notations In this paper an ℓ_p-norm of vector $\mathbf{v} \in \mathbb{R}^n$ is defined as $\|\mathbf{v}\|_p = \left(\sum_{i=1}^{n} |v_i|^p\right)^{\frac{1}{p}}$, where v_i is the i-th element of the vector $\mathbf{v}$. The transpose of $\mathbf{X}$ is denoted as $\mathbf{X}^T$, the inverse of $\mathbf{X}$ is denoted as $\mathbf{X}^{-1}$, and the trace operator of a matrix is denoted as the symbol "tr".

3 Approach

In this section, we first discuss learning the bases of the data. Then we give details on the proposed JGSC for reconstructing the data into the basis space spanned by the learnt bases. We repeat the two processes until the proposed objective function satisfies pre-defined conditions. As a result, each data point is mapped into the basis space to generate its new representation. Finally, we describe how to perform feature selection on the new representation of the data.

3.1 Basis extraction In this subsection, we briefly review the process of learning the bases of the data, aiming at describing the data with a high-level feature representation, i.e., the bases. Such a representation has been shown to make the learning task easier and to obtain much better results in practice [7]. In this paper, a dictionary learning method is used to learn the bases.

Given a data matrix $\mathbf{X} = (x_{ij}) \in \mathbb{R}^{d\times n}$ (where each column represents a data point), we want to learn m bases (or dictionaries) $\mathbf{B} \in \mathbb{R}^{d\times m}$ with the given sparse codes $\mathbf{S} \in \mathbb{R}^{m\times n}$, then the objective function is defined as

$$(3.1)\ \min_{\{\mathbf{B},\mathbf{S}\}} \|\mathbf{X} - \mathbf{B}\mathbf{S}\|_F^2 + \lambda \sum_{i=1}^{n} \|\mathbf{S}_i\|_1, \text{s.t.} \sum_{i=1}^{n}\sum_{j=1}^{d} b_{i,j}^2 \leq 1$$

where $\|.\|_F$ means the Frobenius norm. $\sum_{i=1}^{n}\sum_{j=1}^{d} b_{i,j}^2 \leq 1$ (where $b_{i,j}$ is the element in the i-th row and j-th column of $\mathbf{B}$) is to prevent $\mathbf{B}$ from having arbitrarily large values which would lead to very small values of $\mathbf{S}$. We use the package SPAMS [8] to learn the bases $\mathbf{B}$. The value of $\mathbf{S}$ is obtained in the next subsection.

3.2 Data reconstruction In this subsection we focus on devising a novel and effective model to reconstruct the data into the basis space. To this end, we consider three objectives simultaneously, i.e., minimizing the reconstruction error, preserving the local structures of the data and generating the row sparsity.

Given a set of n data points $\mathbf{X}$ and the learnt bases $\mathbf{B}$, the reconstruction process, which uses $\mathbf{B}$ to reconstruct $\mathbf{X}$ to obtain $\mathbf{X}$'s new representation $\mathbf{S}$, can

be achieved by the following least square loss function

$$\min_{\mathbf{S}} \|\mathbf{X} - \mathbf{BS}\|_F^2 \tag{3.2}$$

where $\mathbf{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_n]$ are the new representation of $\mathbf{X}$.

Given a loss function (Eq.3.2) during optimization, a regularizer is often used to avoid the issue of overfitting as well as to meet predefined criteria such as leading to the sparsity. In this paper, to conduct feature selection, we should choose a regularizer to distinguish the features, i.e., discriminating the important features and the unimportant features. Motivated by the characteristics of the sparsity in sparse coding, we expect the important features to be represented by non-zero values and the unimportant features by zeros after the reconstruction process. Then we discard the unimportant features (i.e., the features with zero values) and use the important features via performing feature selection during the learning process. However, as mentioned before, the ℓ_1-norm regularizer leads to the element sparsity. Instead, the $\ell_{2,1}$-norm regularizer has been designed to measure the distance in feature dimensions via the ℓ_2-norm, while performing summation over different data points via the ℓ_1-norm [10]. Thus the $\ell_{2,1}$-norm regularizer leads to the row sparsity as well as to consider the correlations of all the features. In this paper the $\ell_{2,1}$-norm, as the second goal in the proposed objective function, is proposed as follows

$$\|\mathbf{S}\|_{2,1} = \sum_{j=1}^{m} \|(\mathbf{S})^j\|_2 \tag{3.3}$$

where $(\mathbf{S})^j$ is the j^{th} row of matrix $\mathbf{S}$, which indicates the effect of the j^{th} feature to all the data points.

To effectively perform feature selection, we reconstruct the data via mapping them into the basis space. We might expect that the local structure of each data point in the original space can be well preserved in the basis space. That is, two close data points in the original space should also be close in the basis space. Manifold learning has been shown to achieve this goal as well as to take the correlations among the data points into account. Following the idea in [1], we build a k-nearest-neighbor graph for each data point to achieve such a similarity preservation, which is the third goal in the proposed objective function.

3.3 Pseudo code of the proposed JGSC By integrating the three goals aforementioned together, we obtain the objective function for reconstructing data as

$$\min_{\mathbf{S}} \|\mathbf{X} - \mathbf{BS}\|_F^2 + \alpha tr(\mathbf{SLS}^T) + \lambda \|\mathbf{S}\|_{2,1} \tag{3.4}$$

Algorithm 1 : The proposed JGSC algorithm.

Input: $\mathbf{X} \in \mathbb{R}^{d \times n}$
repeat
  Learn the bases $\mathbf{B}$ from $\mathbf{X}$; See Sec. 3.1.
  Generate $\mathbf{S}$ of $\mathbf{X}$ by Algorithm 2; See Sec. 3.2
until satisfying the predefined conditions

where $\alpha \geq 0$ and $\lambda \geq 0$ are the tuning parameters, and $\mathbf{L}$ is a Laplacian matrix obtained by the built k-nearest-neighbor graph.

In Eq.3.4, the first two terms are designed to address the third problem of MCFS discussed in Section 1, i.e., simultaneously considering the regression process (via the first term) and manifold learning (via the second term) to perform feature selection. The last two terms are designed to handle the first two problems of MCFS, i.e., generating the row sparsity (via the $\ell_{2,1}$-norm regularizer), taking the possible correlations among all the features (via the $\ell_{2,1}$-norm regularizer) and the possible correlations among all data points (via the second term) into account.

By integrating Eq.3.1 into Eq.3.4, the overall objective function of JGSC is defined as

$$\min_{\{\mathbf{B},\mathbf{S}\}} \|\mathbf{X}-\mathbf{BS}\|_F^2+\alpha tr(\mathbf{SLS}^T)+\lambda\|\mathbf{S}\|_{2,1}, \text{ s.t.} \sum_{i=1}^{n}\sum_{j=1}^{d} b_{i,j}^2 \leq 1 \tag{3.5}$$

Actually, the optimization issue in Eq.3.5 is non-convex on both $\mathbf{B}$ and $\mathbf{S}$, but is convex for each one while fixing the other one. Thus we can optimize $\mathbf{S}$ (or $\mathbf{B}$) by fixing $\mathbf{B}$ (or $\mathbf{S}$). We repeat the two steps until the pre-defined conditions are satisfied, e.g., the difference of the values of the objective function in Eq.3.4 between two sequential iterations reaches a given threshold. We summarize the pseudo code of the proposed JGSC in Algorithm 1.

3.4 Features selection After the calculation discussed in Section 3.3, we obtain a new representation $\mathbf{S}$ of the data $\mathbf{X}$. Due to the $\ell_{2,1}$-norm regularizer, many rows in $\mathbf{S}$ shrink to zeros. This indicates that the corresponding features (i.e., these zero rows) are not important to the new representation. To achieve efficiency and effectiveness, we may remove them to perform feature selection. More specifically, we first rank the rows in $\mathbf{S}$ in descending order according to the ℓ_2-norm values of each individual row $\|\mathbf{s}^j\|_2, j = 1, \ldots, m$, and then select top-ranked rows as the results of feature selection.

4 Optimization

The objective function in Eq.3.4 is convex, so it admits the global optimum. However, $\|\mathbf{S}\|_{2,1}$ in Eq.3.4 is

Algorithm 2 : The TOSC algorithm.

Input: $\mathbf{X} \in \mathbb{R}^{d\times n}, \mathbf{B} \in \mathbb{R}^{d\times m}, \mathbf{L} \in \mathbb{R}^{d\times d}, \alpha$ and λ;
Initialize $t = 0$;
$\mathbf{C}_0$ as an $m \times m$ identity matrix;
repeat
  Update $\mathbf{S}_{t+1}$ in Eq.4.6 by Algorithm 3;
  Update $\mathbf{C}_{t+1}$ via Eq.4.7;
  $t = t+1$;
until the objective function in Eq.3.4 converges

convex but non-smooth. In this section we discuss an algorithm to optimize the objective function in Eq.3.4 via calculating the gradient of $\|\mathbf{S}\|_{2,1}$. Then we prove that the proposed algorithm makes the objective function in Eq.3.4 converge to its global optimum.

4.1 The proposed algorithm By setting the derivative of the objective function in Eq.3.4 with respect to $\mathbf{S}$ as zero, we obtain

$$(\mathbf{B}^T\mathbf{B} + \lambda\mathbf{C})\mathbf{S} + \mathbf{S}(\alpha\mathbf{L}) = \mathbf{B}^T\mathbf{X} \tag{4.6}$$

where $\mathbf{C}$ is a diagonal matrix with its i^{th} diagonal element calculated as

$$c_{i,i} = \frac{1}{2\|(\mathbf{S})^i\|_2} \tag{4.7}$$

where $(\mathbf{M})^j$ denotes the i^{th} row of matrix $\mathbf{M}$.

By observing Eq.4.6, we know that $\mathbf{C}$ depends on the value of $\mathbf{S}$; and $\mathbf{S}$ also depends on the value of $\mathbf{C}$. Hence it is impractical to compute $\mathbf{S}$ (or $\mathbf{C}$) directly. In this paper we design a novel iterative algorithm to optimize Eq.4.6 by alternatively computing $\mathbf{S}$ and $\mathbf{C}$ (i.e., an iTerative algOrithm to optimize $\mathbf{S}$ and $\mathbf{C}$, TOSC for short). We first summarize the details in Algorithm 2, and then prove that in each iteration the updated $\mathbf{S}$ and $\mathbf{C}$ make the value of the objective function in Eq.3.4 decrease. As shown in Algorithm 2, in each iteration, given a fixed $\mathbf{C}$, the value of $\mathbf{S}$ is first calculated using Eq.4.6. Then $\mathbf{C}$ is updated using Eq.4.7. The iteration process is repeated until there is no change to the value of the objective function.

Given a fixed $\mathbf{S}$, to solve $\mathbf{C}$ is easy according to Eq.4.7. Given a fixed $\mathbf{C}$, the optimization process for solving $\mathbf{S}$ admits an analytical solution. We denote the proposed Analytical Solution to Solve $\mathbf{S}$ with a fixed $\mathbf{C}$ as the AS3 algorithm[1]. We describe the details of the proposed algorithm as follows and give its pseudo code in Algorithm 3.

Algorithm 3 : The AS3 algorithm.

Input: $\mathbf{X}, \mathbf{B}, \mathbf{L}, \mathbf{C}, \alpha$ and λ;
Obtain $\mathbf{P}$ and $\mathbf{R}$ in Eq.4.8 by Cholesky factorization;
Conduct SVD on $\mathbf{P}$ and $\mathbf{R}$;
Obtain $\tilde{\mathbf{\Sigma}}_1$, $\tilde{\mathbf{\Sigma}}_2$, $\tilde{\mathbf{S}}$ and $\mathbf{Q}$ according to Eq.4.12;
Obtain $\tilde{\mathbf{S}}_{i,j}$ according to Eq.4.13;
Obtain $\mathbf{S}$ according to Eq.4.15;

Since both $\mathbf{B}^T\mathbf{B} + \lambda\mathbf{C}$ and $\alpha\mathbf{L}$ are systemic and positive-definite, we perform Cholesky factorization on them respectively to produce lower triangular matrices $\mathbf{P}$ and $\mathbf{R}$ to satisfy the equations:

$$\begin{aligned} \mathbf{B}^T\mathbf{B} + \lambda\mathbf{C} &= \mathbf{P}^T \times \mathbf{P} \\ \alpha\mathbf{L} &= \mathbf{R} \times \mathbf{R}^T \end{aligned} \tag{4.8}$$

By performing eigen-decomposition on $\mathbf{P}$ and $\mathbf{R}$ respectively, we denote

$$\begin{aligned} \mathbf{P} &= \mathbf{U}_1\mathbf{\Sigma}_1\mathbf{V}_1^T \\ \mathbf{R} &= \mathbf{U}_2\mathbf{\Sigma}_2\mathbf{V}_2^T \end{aligned} \tag{4.9}$$

where both $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices. Then Eq.4.6 can be expressed as

$$\mathbf{V}_1\mathbf{\Sigma}_1^T\mathbf{\Sigma}_1\mathbf{V}_1^T\mathbf{S} + \mathbf{S}\mathbf{U}_2\mathbf{\Sigma}_2\mathbf{\Sigma}_2^T\mathbf{U}_2^T = \mathbf{B}^T\mathbf{X} \tag{4.10}$$

By multiplying $\mathbf{V}_1^T$ and $\mathbf{U}_2$ from the left and the right on both sides of Eq.4.10 respectively, we obtain

$$\mathbf{\Sigma}_1^T\mathbf{\Sigma}_1\mathbf{V}_1^T\mathbf{S}\mathbf{U}_2 + \mathbf{V}_1^T\mathbf{S}\mathbf{U}_2\mathbf{\Sigma}_2\mathbf{\Sigma}_2^T = \mathbf{V}_1^T\mathbf{B}^T\mathbf{X}\mathbf{U}_2 \tag{4.11}$$

By denoting

$$\begin{aligned} \mathbf{\Sigma}_1^T\mathbf{\Sigma}_1 &= \tilde{\mathbf{\Sigma}}_1 = \mathrm{diag}(\sigma_1^{(1)}, ..., \sigma_1^{(d)}) \\ \mathbf{\Sigma}_2\mathbf{\Sigma}_2^T &= \tilde{\mathbf{\Sigma}}_2 = \mathrm{diag}(\sigma_2^{(1)}, ..., \sigma_2^{(m)}) \\ \mathbf{V}_1^T\mathbf{S}\mathbf{U}_2 &= \tilde{\mathbf{S}} \\ -\mathbf{V}_1^T\mathbf{B}^T\mathbf{X}\mathbf{U}_2 &= \mathbf{Q} \end{aligned} \tag{4.12}$$

and denoting "*diag*" as the diagonal operation, Eq.4.11 becomes

$$\tilde{\mathbf{\Sigma}}_1\tilde{\mathbf{S}} + \tilde{\mathbf{S}}\tilde{\mathbf{\Sigma}}_2 = \mathbf{Q} \tag{4.13}$$

Thus the elements in $\tilde{\mathbf{S}}$ can be obtained by

$$\tilde{\mathbf{S}}_{i,j} = \frac{\mathbf{Q}}{\sigma_1^i + \sigma_2^j} \tag{4.14}$$

After getting the $\tilde{\mathbf{S}}$, we can obtain the optimal $\mathbf{S}$ as

$$\mathbf{S} = \mathbf{V}_1\tilde{\mathbf{S}}\mathbf{U}_2^T \tag{4.15}$$

[1]We should state that the equation to optimize $\mathbf{S}$ with fixed $\mathbf{C}$ in Eq.4.7 is called Sylvester equation, which can be solved by directly employing Matlab function lyap or software LAPACK with complexity $O(n^3)$, n is sample size. Such a high complexity makes Sylvester equation unfeasible in large-scale datasets. In this paper the proposed AS3 algorithm solves Sylvester equation by an analytical solution with complexity is $min(n^2d, d^3)$, d is the dimensionality. This makes Sylvester equation to be applied in large-scale datasets.

4.2 Convergence In this subsection we show that the proposed Algorithm 2 makes the value of the objective function in Eq.3.4 monotonically decrease via Theorem 4.1 below. We first give a lemma as follows.

LEMMA 4.1. *For any positive values a_i and b_i, $i = 1, ..., m$, the following holds:*

$$\sum_{i=1}^{m} \frac{b_i^2}{a_i} \leq \sum_{i=1}^{m} \frac{a_i^2}{a_i} \Longleftrightarrow \sum_{i=1}^{m} \frac{(b_i+a_i)(b_i-a_i)}{a_i} \leq 0$$

$$\Longleftrightarrow \sum_{i=1}^{m}(b_i - a_i) \leq 0 \Longleftrightarrow \sum_{i=1}^{m} b_i \leq \sum_{i=1}^{m} a_i \quad (4.16)$$

THEOREM 4.1. *With the Lemma 4.1 and following the proof in the literature [10], we can easily prove that in each iteration Algorithm 2 monotonically decreases the objective function value in Eq.3.4.*

4.3 Discussion As mentioned in Section 1, the objective function in the algorithms (such as MRSF, GSC and our JGSC) employ least square loss function, but use different regularizers. For example, MRSF utilizes $\ell_{2,1}$-norm regularizer, GSC uses ℓ_1-norm regularizer, and the proposed JGSC utilizes two regularizers, i.e., $\ell_{2,1}$-norm and Laplacian matrix of data respectively.

The algorithm STDR [18] employs same regularizers as our JGSC but utilizes robust loss function[2]. Moreover, STDR learns the bases of training data from external data while JGSC obtains them from training data because STDR assumes learning with limited training data and JGSC does not make such assumption. More specifically, STDR belongs to self-taught learning but JGSC is unsupervised learning. The most distinguished thing between STDR and JGSC is the solution of Sylvester equation with low complexity. STDR employs Matlab function lyap but our JGSC proposes an analytical solution. Details please see Footnote 1.

5 Experiment analysis

In order to evaluate the performance of the proposed JGSC model, we compare it with several state-of-the-art dimensionality reduction algorithms in terms of classification accuracy on three types of real applications, including text mining (with datasets PCMAC and BASEHOCK), face recognition (AR10P and PIE10P) and bioinformatics (TOX and SMK-CAN).

[2]In our experiments we found no significant difference between robust loss function and least square loss function in our proposed framework. We did not report the results in the experimental part because we do not focus on this in this paper.

5.1 Experiment setup The used datasets are downloaded from http://featureselection.asu.edu/datasets.php and http://www.zjucadcg.cn/dengcai/Data/data.html respectively. The number of the features in the used datasets varies from 2,420 to 19,993.

We compare JGSC with the following algorithms, "Original" (i.e., using all original features to perform *kNN* classification), MCFS [2], MRSF [15], GSC [3, 16], and UDFS [13], which is an embedded approach via taking the $\ell_{2,1}$-norm regularizer and discriminative ability into account.

In our experiments, we set the values of parameters for the comparison algorithms according to the instructions in their papers. We perform line search with 4 levels for each parameter in our JGSC. In all algorithms, the number of left dimensionality is kept as $\{10\%, 20\%, ..., 80\%\}$ of those of the original ones for each dataset. The number of k in the *kNN* classifier is set as 5. Given a reduced dataset, we perform 10-fold cross validation with the *kNN* algorithm. The average results among ten runs and the corresponding standard deviation are reported in this paper.

In the following subsections, we first evaluate convergence rate of the proposed JGSC on all datasets, for evaluating the efficiency of our proposed optimization Algorithm 2, in terms of the objective function value in each iteration. Then we test the parameters' sensitivity of the proposed JGSC according to the variation of the parameters, such as λ and α in Eq.3.4, aiming at achieving the best performance of the proposed JGSC. Finally, we compare JGSC with the comparison algorithms in terms of average classification accuracy (ACA for short).

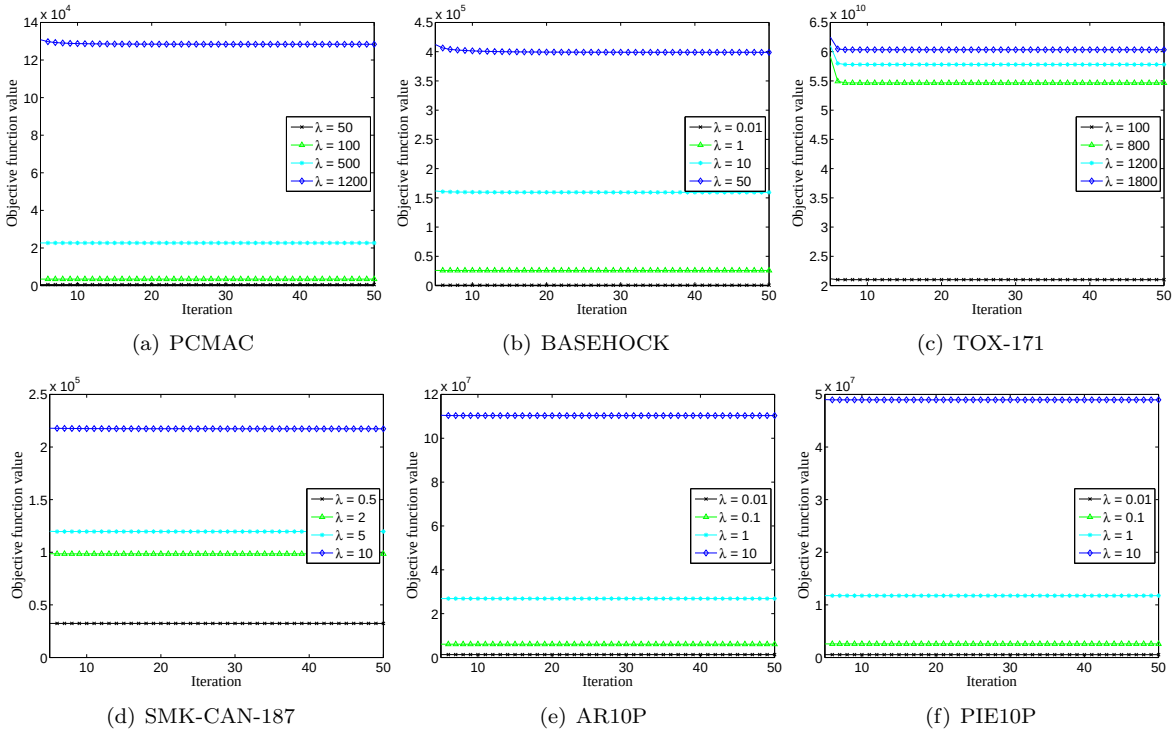
Given a data point $\mathbf{x}_i$, let y_i and $\hat{y}_i$ be the derived labels via a classification algorithm and the true label respectively. The ACA is defined as follows:

$$ACA = \frac{\sum_{i=1}^{n} \delta(y_i, \hat{y}_i)}{n} \quad (5.17)$$

where n is the sample size, $\delta(x, y) = 1$ if $x = y$, and $\delta(x, y) = 0$, otherwise. The larger the performance on ACA is, the better the method.

5.2 Convergence rate In this subsection, we want to know the convergence rate of Algorithm 2, so we report some of the results in Fig.3 due to lack of space. Fig.3 shows the results of the objective function value while fixing the value of α and varying λ.

We can observe in Fig.3: 1) the objective function value rapidly decreases at the first few iterations; and 2) the objective function value becomes stable after about 30 iterations (or even less than 20 iterations in



(a) PCMAC (b) BASEHOCK (c) TOX-171

(d) SMK-CAN-187 (e) AR10P (f) PIE10P

Figure 3: An illustration on convergence rate of Algorithm 2 for solving the proposed objective function with fixed α.

many cases) on all datasets. This confirms that the fast convergence rate of Algorithm 2 to solve the proposed optimization problem in Eq.3.4. Similar results are observed for other α and λ values.

5.3 Parameters' sensitivity In this subsection we study the classification performance of JGSC with respect to the variations of different parameters' settings, i.e., λ and α in Eq.3.4. Due to the page limit, we only report the results on the case with 50% dimensionality left from the original dimensionality. The other cases have similar results. The experimental results are presented in Fig. 4.

As can be seen in Fig. 4, JGSC is sensitive to the parameter setting. According to the experimental results, the average maximal improvement of the best results to the worst ones in the six datasets varies from 17.76% to 85.38%. Moreover, to obtain the best classification performance, different datasets need to set parameters differently.

5.4 Classification results by all algorithms The classification performance ACA by all algorithms is presented in Fig. 5. The horizontal axis represents the number of the dimensions left after performing feature selection, and the values vary from 10% to 80%. The vertical axis describes the value of ACA.

According to Fig.5, we have the following observations: 1) The proposed JGSC achieves the best performance. This is obviously because the proposed JGSC overcomes all three drawbacks of MCFS, but each of the rival algorithms only solves part of the problems. Moreover, JGSC outperforms UDFS, which also considers two constraints (i.e., discriminative ability and manifold learning with the $\ell_{2,1}$-norm regularizer) to perform feature selection. 2) The results of "Original" are better than the results of some dimensionality reduction algorithms on some datasets. However, these dimensionality reduction algorithms are more efficient due to their significantly reduced dimensionality of the datasets. Thus it is crucial for conducting dimensionality reduction on high-dimensional data. This is consistent with the conclusion in [2, 4, 13]. 3) Although both MCFS and GSC lead to the element sparsity, the classification performance of MCFS is worse than GSC. This occurs because GSC simultaneously performs manifold learning and regression, but MCFS sequentially performs them. This demonstrates that simultaneously performing manifold learning and regression is better. 4) MRSF outperforms MCFS. That is because MRSF has more advantages

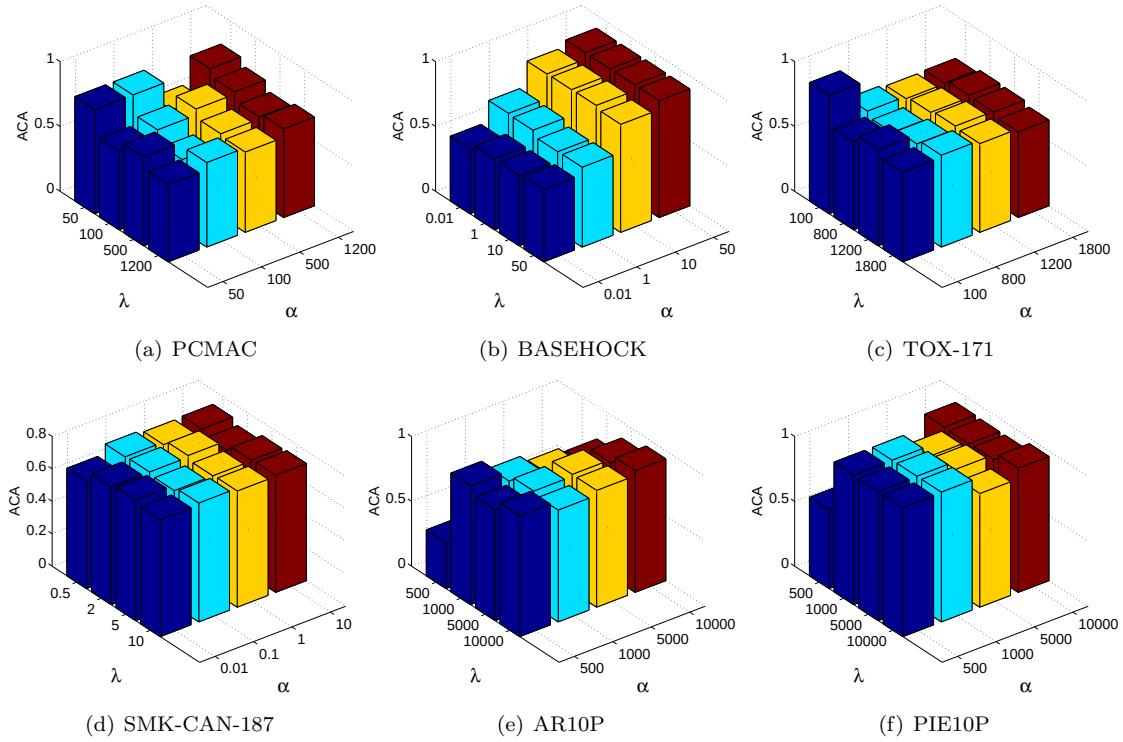


Figure 4: ACA results on different datasets with different parameters.

than MCFS by replacing the ℓ_1-norm regularizer with the $\ell_{2,1}$-norm regularizer, such as considering the correlations among the spectral features and leading to the row sparsity. 5) The literatures (e.g., [2, 13]) have shown that their methods (e.g., UDFS and MCFS) outperform the popular methods, such as LPP [5], LScore [4], so we can make the conclusion that JGSC also outperforms LPP and LScore.

6 Conclusion

In this paper, we propose a novel feature selection method to deal with high-dimensional data. Moreover, we have shown that our method theoretically makes the proposed objective function converge to its global optimum. Furthermore, the experimental results have shown the effectiveness of the proposed method. In the future we will extend the proposed method into its kernel edition to deal with more complex cases, such as datasets with limited data points.

7 Acknowledgements

This research has been supported by the US National Science Foundation (NSF) under grant CCF-0905337; the Australian Research Council (ARC) under large grant DP0985456; the Nature Science Foundation (NSF) of China under grants 61170131 and 61263035; the China 863 Program under grant 2012AA011005; the China 973 Program under grant 2013CB329404; the Guangxi Natural Science Foundation under grant 2012GXNSFGA060004; and the Guangxi "Bagui" Teams for Innovation and Research.

References

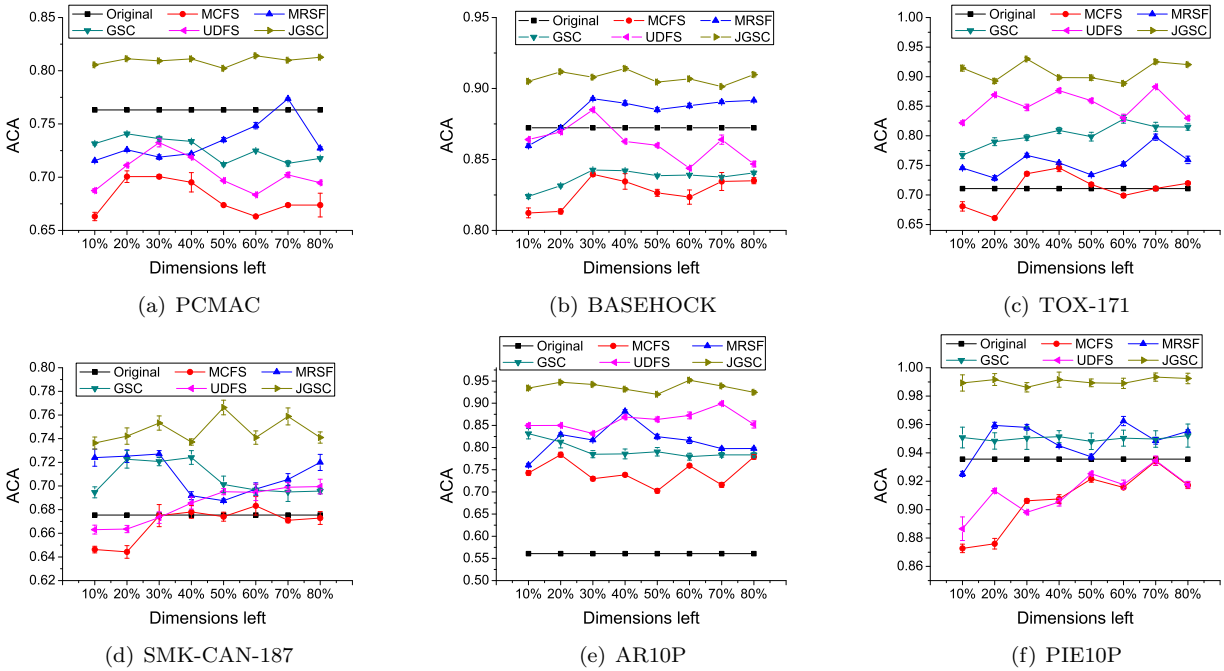
[1] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

[2] Deng Cai, Chiyuan Zhang, and Xiaofei He. Unsupervised feature selection for multi-cluster data. In *ACM Special Interest Group on Knowledge Discovery and Data Mining*, pages 333–342, 2010.

[3] Shenghua Gao, Ivor Wai-Hung Tsang, Liang-Tien Chia, and Peilin Zhao. Local features are not lonely - laplacian sparse coding for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3555–3561, 2010.

[4] Xiaofei He, Deng Cai, and Partha Niyogi. Laplacian score for feature selection. In *Neural Information Processing Systems*, pages 1–8, 2005.

[5] Xiaofei He and Partha Niyogi. Locality preserving pro-



(a) PCMAC (b) BASEHOCK (c) TOX-171

(d) SMK-CAN-187 (e) AR10P (f) PIE10P

Figure 5: The results of ACA on various parameters' settings on different datasets. Note that, the range shown at the curves represents the performance standard deviation.

jections. In *Neural Information Processing Systems*, pages 197–204, 2003.

[6] Sotiris Kotsiantis. Feature selection for machine learning classification problems: a recent overview. *Artificial Intelligence Review*, pages 1–20, 2011.

[7] Honglak Lee, Rajat Raina, Alex Teichman, and Andrew Y. Ng. Exponential family sparse coding with applications to self-taught learning. In *International Joint Conference on Artificial Intelligence*, pages 1113–1119, 2009.

[8] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11:19–60, 2010.

[9] Sebastián Maldonado and Richard Weber. A wrapper method for feature selection using support vector machines. *Information Science Journal*, 179:2208–2217, 2009.

[10] Feiping Nie, Heng Huang, Xiao Cai, and Chris Ding. Efficient and robust feature selection via joint l2,1-norms minimization. In *Neural Information Processing Systems*, pages 1813–1821, 2010.

[11] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. Self-taught learning: transfer learning from unlabeled data. In *International Conference on Machine Learning*, pages 759–766, 2007.

[12] Zenglin Xu, Rong Jin, Jieping Ye, and Michael R. Lyu. Discriminative semi-supervised feature selection via manifold regularization. *IEEE Transactions on Neural Networks*, 21(7):1033–1047, 2010.

[13] Yi Yang, Heng Tao Shen, Zhigang Ma, Zi Huang, and Xiaofang Zhou. $\ell_{2,1}$-norm regularized discriminative feature selection for unsupervised learning. In *International Joint Conferences on Artificial Intelligence*, pages 1589–1594, 2011.

[14] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 68:49–67, 2006.

[15] Zheng Zhao, Lei Wang, and Huang Liu. Efficient spectral feature selection with minimum redundancy. In *AAAI Conference on Artificial Intelligence*, 2010.

[16] Miao Zheng, Jiajun Bu, Chun Chen, Can Wang, Lijun Zhang, Guang Qiu, and Deng Cai. Graph regularized sparse coding for image representation. *IEEE Transactions on Image Processing*, 20(5):1327 – 1336, 2011.

[17] Xiaofeng Zhu, Zi Huang, Heng Tao Shen, Jian Cheng, and Changsheng Xu. Dimensionality reduction by mixed kernel canonical correlation analysis. *Pattern Recognition*, 45(8):3003–3016, 2012.

[18] Xiaofeng Zhu, Zi Huang, Yang Yang, Heng Tao Shen, Changsheng Xu, and Jiebo Luo. Self-taught dimensionality reduction on the high-dimensional small-sized data. *Pattern Recognition*, 46(1):215–229, 2013.