Graph-Based Learning via Auto-Grouped Sparse Regularization and Kernelized Extension

Yuqiang Fang, Ruili Wang, Bin Dai, and Xindong Wu, *Fellow, IEEE*

Abstract—The key task in developing graph-based learning algorithms is constructing an informative graph to express the contextual information of a data manifold. Since traditional graph construction methods are sensitive to noise and less datum-adaptive to changes in density, a new method called ℓ^1-graph was proposed recently. A graph construction needs to have two important properties: sparsity and locality. The ℓ^1-graph has a strong sparsity property, but a weak locality property. Thus, we propose a new method of constructing an informative graph using auto-grouped sparse regularization based on the ℓ^1-graph, which is called as Group Sparse graph (GS-graph). We also show how to efficiently construct a GS-graph in reproducing kernel Hilbert space with the kernel trick. The new methods, the GS-graph and its kernelized version (KGS-graph), have the same noise-insensitive property as that of ℓ^1-graph and also can successively preserve the properties of sparsity and locality simultaneously. Furthermore, we integrate the proposed graph with several graph-based learning algorithms to demonstrate the effectiveness of our method. The empirical studies on benchmarks show that the proposed methods outperform the ℓ^1-graph and other traditional graph construction methods in various learning tasks.

Index Terms—Graph based learning, sparse representation, spectral embedding, subspace learning, non-negative matrix factorization

1 Introduction

In recent years, manifold-based learning has become an emerging and promising method in machine learning, with numerous applications in data analysis including dimensionality reduction [1], [2], [3], [4], [5], clustering [2], [6], [7] and classification [8], [9]. The main assumption in these dimensionality reduction and classification methods is that the data resides on a low dimensional manifold embedded in a higher dimensional space. When approximating the underlying manifold, the most common strategy is to construct an informative graph. The graph can be viewed as a discretized approximation of a manifold sampled by the input patterns. There are many manifold learning based dimensionality reduction algorithms. For example, Isometric Feature Mapping (ISOMAP) [1], a widely used manifold embedding method, extends metric multidimensional scaling by incorporating the geodesic distances of all pairs of measurements imposed by a global weighted graph. Laplacian Eigenmaps (LE) [2] and Locally Linear Embedding (LLE) [3] preserve proximity relationships through data manipulations on an undirected weighted graph that indicates the neighbor relations of pairwise measurements. Manifold-based clustering, e.g., spectral clustering, also can be solved by graph partitioning. Moreover, manifold subspace learning, e.g., Locality Preserving Projections (LPP) [4] and Neighborhood Preserving Embedding (NPE) [5], can be explained in a general graph framework [10]. We can see that graph plays a key role in these graph-based learning algorithms.

In most graph-based learning methods, a graph is constructed by calculating pairwise euclidean distances, e.g., k-nearest neighbor graph. However, a graph based on pairwise distances is very sensitive to unwanted noise. To handle such problem, recently, a new method (so called ℓ^1-graph [9]) was proposed, which constructs the graph based on a modified sparse representation framework [11], [12]. Sparse representation-based linear projections (SRLP) [13] and sparsity preserving projections (SPP) [7] were two new subspace learning methods that have a similar idea with that of the ℓ^1-graph. They all choose local neighborhood information for dimensionality reduction by minimizing the ℓ^1-regularization objective function. Although it has shown that the ℓ^1-graph based algorithms [7], [9], [13] outperform principle component analysis (PCA) [14], LPP [4] and NPE [5] on several data sets, the ℓ^1-graph based algorithms only have the sparsity property, but do not have the locality property (more details can be seen in Section 3.1).

In this paper, based on ℓ^1-graph, we propose a new method to build a graph that has both sparsity and locality properties. As we know, high-dimensional data often observe sparsity and locality, which should be taken into account in graph-based learning [2]. However, in ℓ^1-graph, the regularization term of ℓ^1-norm tends to select few bases for graph construction to be in favor of sparsity, thus losing the locality property. Motivated by the above observations, we introduce two sparse regularization methods (Elastic net [15] and octagonal shrinkage and clustering algorithm for regression (OSCAR) [16]) that

- *Y. Fang and B. Dai are with the College of Mechatronic Engineering and Automation, National University of Defense Technology, Changsha, 410073 P.R. China. E-mail: {fangyuqiang, daibin}@nudt.edu.cn.*
- *R. Wang is with the School of Engineering and Advanced Technology, Massey University, Albany, Auckland, New Zealand. E-mail: r.wang@massey.ac.nz.*
- *X. Wu is with the School of Computer Science and Information Engineering, Hefei University of Technology, China and the Department of Computer Science, University of Vermont, Burlington, Vermont. E-mail: xwu@uvm.edu.*

Manuscript received 31 May 2013; revised 27 Jan. 2014; accepted 20 Feb. 2014. Date of publication 19 Mar. 2014; date of current version 1 Dec. 2014. Recommended for acceptance by R. Jin.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2014.2312322

have the automatic group effect for the graph construction. Then a novel group sparse graph method (GS-graph) and its kernelized version (KGS-graph) are proposed for several graph-based learning algorithms. The proposed graph has the same noise-insensitive property as that of ℓ^1-graph, and also has successively preserved the group and local information in the graph. Our empirical studies on benchmark data sets demonstrate the promising results of the proposed method.

The rest of this paper is organized as follows. In Section 2, the related work on graph-based learning algorithms is reviewed. In Section 3, the main disadvantage of sparse graph construction is analyzed and then our newly developed group sparse based graph construction algorithm and its kernelized version are introduced for several graph-based learning algorithms. In Section 4, the experimental results and analysis are then presented. Finally, conclusions and discussions are provided and the further work is also indicated at the end of the last section. Note that the preliminary result of this research has been reported in a conference paper in [17]. Compared with the paper [17], we further improve our work in the following aspects: (1) Motivated by the success of the kernel trick in capturing the nonlinear similarity of features, we propose the KGS-graph construction method in this paper, which extends our original method by an implicit kernel function; (2) We perform more extensive experiments to compare our GS-graph and KGS-graph with related graph construction methods in data clustering and classification tasks; (3) We also show more comprehensive theoretical analysis for our method, including the algorithm details, computational complexity and the proof of the locality property.

Notation. For any vector $\boldsymbol{x}$, its transpose is denoted by $\boldsymbol{x}^\top$, and its ith component is $\boldsymbol{x}[i]$. The ℓ^1-norm of $\boldsymbol{x}$ is $\|\boldsymbol{x}\|_1 = \sum_i |\boldsymbol{x}[i]|$, and its ℓ^2-norm is $\|\boldsymbol{x}\|_2 = \sqrt{\sum_i (\boldsymbol{x}[i])^2}$. For any matrix $\boldsymbol{X} \in \mathbb{R}^{n\times n}$, we use $\sigma_{max}(\boldsymbol{X})$ to denote the largest eigenvalue of $\boldsymbol{X}$. Moreover, $(\cdot)_+ = \max(0, \cdot)$ is the thresholding function and $\mathrm{sgn}(\cdot)$ is the signum function.

2 Related Work

2.1 Graph-Based Learning

Although the motivations of different manifold and graph-based learning algorithms vary, their objectives are similar, which are to derive a lower-dimensional manifold representation of high-dimensional data, to facilitate related tasks. The central to them is constructing a graph structure that models the contextual information (i.e., geometrical and discriminative structure) of the data manifold.

Suppose we have n data points represented as a matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]$, where $\boldsymbol{x}_i \in \mathbb{R}^m$. With the data points, we can build a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the vertex set of the graph is referred as $\mathcal{V}(\mathcal{G}) = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$, its edge set to $\mathcal{E}(\mathcal{G}) = \{e_{ij}\}$. The number of vertices of a graph $\mathcal{G}$ is its *order* [18], written as $|\mathcal{G}|$; its number of edges is denoted by $\|\mathcal{G}\|$. If an edge e_{ij} connects vertices $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, we denote the relation as $i \sim j$. The number of neighbors of a node $\boldsymbol{x}$ is called *degree* of $\boldsymbol{x}$ and is denoted by $d_{\mathcal{G}}(\boldsymbol{x})$, $d_{\mathcal{G}}(\boldsymbol{x}_i) = \sum_{i\sim j} e_{ij}$. Further, each edge e_{ij} can be weighted by $w_{ij} > 0$ for pairwise similarity measurements.

For the above notation, it is easy to see that edge e_{ij} and weight w_{ij} are important factors in graph construction. In common graph-based learning algorithms, the edges and weights are often specified in the following manners:

Global graph. For $\forall i, j,\ i \sim j,\ w_{ij} = f(\mathrm{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j))$[1] and $d(\boldsymbol{x}_i) = n - 1$;

kNN graph. For $\forall i,\ i \sim k,\ \boldsymbol{x}_k$ belongs to the k-nearest neighbor vertices for $\boldsymbol{x}_i$, $w_{ik} = f(\mathrm{dist}(\boldsymbol{x}_i, \boldsymbol{x}_k))$, $d(\boldsymbol{x}_i) = k$;

ε NN graph. For $\forall i,\ i \sim j$, if $\mathrm{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j) \le \varepsilon$, $w_{ij} = f(\mathrm{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j))$ and $d(\boldsymbol{x}_i)$ is $k_\varepsilon(\boldsymbol{x}_i)$[2];

Moreover, to describe the concept of sparse graph, we induce the definition of *graph density* as follow:

Definition 1 (Graph Density [19]). *For an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the graph density of $\mathcal{G}$ is $\frac{2\|\mathcal{G}\|}{|\mathcal{G}|(|\mathcal{G}|-1)}$.*

From the above definitions, one can see that the density of global graph is 1 and the kNN graph has a low density $\frac{k}{n-1}$ when $k \ll n$. Formally, we define a *dense graph* is a graph which has a high graph density, while a graph with a low graph density is a *sparse graph*.

The sparse graph plays an important role in graph-based learning. From the sparse graph, one can construct a matrix whose spectral decompositions reveal the low dimensional structure of the submanifold [2]. Thus, with an appropriate sparse graph, we can set up a quadratic objective function derived from the graph for embedding or subspace learning and solved by the eigenvectors of eigen-problem [10]. Also, the sparse graph based function can be incorporated as a geometric regularization in semi-supervised learning, transductive inference [8] or non-negative matrix factorization (NMF) [6].

2.2 Learning with ℓ^1-Graph

Since the above traditional graph construction methods are sensitive to data noise and less datum-adaptive to changes in density, a new construct method (so-called ℓ^1-graph) via sparse representation was proposed and harnessed for prevalent graph-based learning tasks [9]. The assumption of ℓ^1-graph is that each datum can be endogenously sparse reconstructed by similar training data. Sparse reconstructive coefficients tend to describe the local neighborhood information, which can be obtained by minimizing an ℓ^1-optimization problem or Lasso problem in Statistics [11].

In Algorithm 1, the identity matrix $\boldsymbol{I}$ is introduced as a part of the dictionary to code the noise, e.g., the corrupted or occluded pixels in an image [12]. That makes ℓ^1-graph more robust to noise than other pairwise graph construction manners. In addition, from Equation (1), we can see that the neighbors of $\boldsymbol{x}_i$ is automatically determined by solving an ℓ^1-regularized linear regression problem. Especially, ℓ^1-regularization as a convex relaxation of ℓ^0-regularization promotes sparsity in the solution $\boldsymbol{\alpha}_i$. Also, this sparse solution determines the set of support samples, which is closest to the given sample $\boldsymbol{x}_i$. Formally, if we define the support of a sparse vector $\boldsymbol{\alpha}_i$ as $supp(\boldsymbol{\alpha}_i) = \{j : \boldsymbol{\alpha}_i[j] \neq 0\}$, the graph

1. $\mathrm{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ denotes the euclidean distance or problem-dependent distance between two vertices; $f(\cdot)$ is the non-increasing weight function, e.g., *Gaussian weight function* $f(x) = \exp(-\frac{x^2}{2\sigma^2})$ or *unit weight function* $f(x) \equiv 1$.

2. $k_\varepsilon(\boldsymbol{x}_i)$ is used to denote the number of node $\boldsymbol{x}$ which satisfies $\mathrm{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j) \le \varepsilon$.

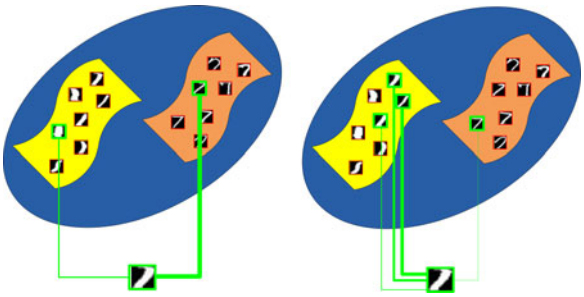


Fig. 1. ℓ^1-norm regularization encourages sparsity, but it tends to select few samples randomly from the entire class. For example, digit "7" is similar to digit "1" in some situations, Lasso may choose a single wrong sample as shown in the left figure. However, group sparse regularization encourages group sparsity, which tries to represent the test sample with a group of similar samples as shown in the right figure.

density of ℓ^1-graph is $\frac{\sum_i^n |supp(\boldsymbol{\alpha}_i)|}{n(n-1)}$. Since $|supp(\boldsymbol{\alpha}_i)| \ll n$, ℓ^1-graph can be defined as a sparse graph. Now, we summarize the ℓ^1-graph construction method as follow:

ℓ^1-graph. For $\forall i$, $i \sim j$, if $\boldsymbol{\alpha}_i^*[j] \neq 0$, $w_{ij} = |\boldsymbol{\alpha}_i^*[j]|$ when $i > j$ and $|\boldsymbol{\alpha}_i^*[j-1]|$ when $i < j$, $d(\boldsymbol{x}_i) = |supp(\boldsymbol{\alpha}_i)|$.

Algorithm 1: ℓ^1-graph Construction [9]

Input : Data matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{m \times n}$
Output: Affinity matrix $\boldsymbol{W} \in \mathbb{R}^{n \times n}$

1 Normalize the data point $\boldsymbol{x}_i$ with $\|\boldsymbol{x}_i\|_2 = 1$;
2 For each data point $\boldsymbol{x}_i$, find sparse coefficient $\boldsymbol{\alpha}_i^*$ and $\boldsymbol{e}_i^*$ from the ℓ^1 norm regularization problem:

$$\min \|\boldsymbol{x}_i - [\boldsymbol{X}_{-i} \ \boldsymbol{I}] \begin{bmatrix} \boldsymbol{\alpha}_i \\ \boldsymbol{e}_i \end{bmatrix} \|_2^2 + \lambda \| \begin{bmatrix} \boldsymbol{\alpha}_i \\ \boldsymbol{e}_i \end{bmatrix} \|_1, \quad (1)$$

where $\boldsymbol{X}_{-i} = [\boldsymbol{X} \setminus \boldsymbol{x}_i] \in \mathbb{R}^{m \times (n-1)}$; $\boldsymbol{I}$ is an m-order identity matrix, $\boldsymbol{\alpha}_i \in \mathbb{R}^{n-1}$;
3 Set affinity matrix $\boldsymbol{W}_{ij} = |\boldsymbol{\alpha}_i^*[j]|$ if $i > j$, $\boldsymbol{W}_{ij} = |\boldsymbol{\alpha}_i^*[j-1]|$ if $i < j$ and $\boldsymbol{W}_{ij} = 0$ if $i = j$.

3 Proposed Method

3.1 Sparsity versus Group Sparsity

Research in manifold or graph-based learning shows that a sparse graph characterizing locality relations can convey the valuable information for classification and clustering [2]. Thus, two of important issues in graph construction are *sparsity* and *locality*.

The ℓ^1-graph just considers the sparse representation during sparse graph construction. One can choose the weights and edges connecting $\boldsymbol{x}_i$ to other vertices by solving the Lasso problem, and utilize the recovery coefficients to reveal the latent locally sparsity. In our opinion, the ℓ^1-graph has the following limitations:

(1) ℓ^1-norm (Lasso) regularization encourages sparsity without any consideration of locality. Indeed, most graph-oriented learning algorithms are proposed under the manifold assumption [1], [2], [3], [4], [5], [8]. Also, the graphs in the learning algorithm are used to approximating the underlying manifold. Furthermore, the core of the manifold concept is *locally euclidean*, equivalents to the requirement that each data point $\boldsymbol{x}_i$ has a neighborhood subspace U homeomorphic to an n-ball in $\mathbb{R}^n$ [20]. Ideally, when

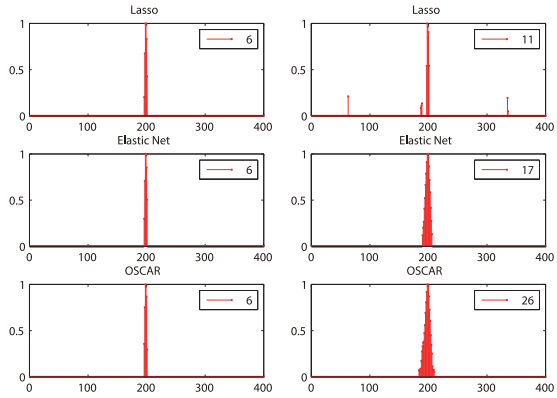


Fig. 2. Nonzero sparse coefficients solved by Lasso, Elastic net and OSCAR; *Left*: Keeping the same number of nonzero coefficients by adjusting regularization parameters; *Right*: Increasing the number of nonzero coefficients to compare the performances of different regularization.

constructing a graph via endogenous sparse representation, we desire the neighborhood subspace U is supported with the data that are indicated by the nonzero sparse coefficients. That means the support samples are highly correlated with each other to satisfy the property *locally Euclidean*. Thus, we desire the nonzero coefficients locality and sparsity not merely sparsity.

(2) ℓ^1-norm regularization encourages sparsity, but it tends to select only one sample from the entire class [15], [21], as a nearest neighbor selector in the extreme case. Thus, when some samples are correlated from different classes (e.g., digit "7" is similar to digit "1" in some situations, but they belong to different classes), Lasso may choose a single wrong sample to represent the test sample as shown in Fig. 1. Thus, ℓ^1-graph is too sparse to keep the high discriminating power for graph-based learning.

(3) Without a group constraint, the nonzero sparse coefficients by ℓ^1-norm regularization tend to unstable and the result is difficult to be interpreted. For example, in Fig. 2, when we adjust the regularization parameters to increase the nonzero sparse coefficients (i.e., the degree of a node), some small weight coefficients solved by Lasso will be randomly distributed[3]. Thus, we need some new group sparsity regularizers, as discussed in next section, which can keep coefficients group and clustered sparse.

In summary, ℓ^1-norm regularization for sparse graph construction has a major limitation, which can only satisfy the *sparsity* constraint, but cannot satisfy the *locality* constraint simultaneously. To overcome this limitation, we introduce and extend two alternate regularization methods that can enforce automatically group sparsity for the graph construction.

3.2 GS-Graph Construction

The problem of group sparsity is studied in [22], [23]. They assume that the sparse coefficients in the same group tend to be either zero or nonzero simultaneously.

3. In this example, we built a dictionary with 400 noised images from the teapot data set (details described in Section 4.1), and then one selected image is reconstructed by different methods, all coefficients are normalized and shown in the figure.

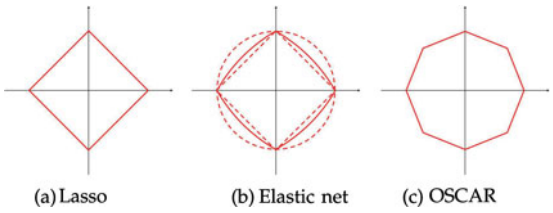


Fig. 3. Graphical representation of the constraint regions of Lasso, Elastic net, and OSCAR; Especially, OSCAR has an octagonal constraint region, which can achieve a grouping property [16]. The Elastic net can be adjusted between ℓ^1- and ℓ^2-regularization with different tuning parameters (dashed lines). Thus, the Elastic net could induce a grouping property with appropriate tuning.

However, in these papers, the label information of groups is required to be known in advance. In other words, they belong to supervised learning. However, in our method, we focus on unsupervised learning, the same as ℓ^1-graph [9]. When constructing a sparse graph in an unsupervised scenario, the label information of groups can be unknown in the data set but the sparsity and group clustering tend are known.

In this paper, two sparse regularization methods with an auto-grouped effect are introduced and extended for graph construction, Elastic net [15] and OSCAR [16].

Elastic net. The Elastic net regularizer is a combination of the ℓ^1- and ℓ^2-norms. The ℓ^1-penalty promotes sparsity, while ℓ^2-penalty encourages the grouping effect [15]. When applying this regularization to constructing a sparse graph, we can modify Equation (1) as follows:

$$\min_{\boldsymbol{\alpha}_i} \frac{1}{2}\left\|\boldsymbol{x}_i - \boldsymbol{X}_{-i}\boldsymbol{\alpha}_i\right\|_2^2 + \lambda_1\|\boldsymbol{\alpha}_i\|_1 + \frac{\lambda_2}{2}\left\|\boldsymbol{\alpha}_i\right\|_2^2, \tag{2}$$

where λ_1 and λ_2 are regularization parameters.

OSCAR. Octagonal shrinkage and clustering algorithm for regression [16] is a novel sparse model that constructs a regularizer with a weighted combination of ℓ^1-norm and a pairwise ℓ^∞-norm. OSCAR encourages both sparsity and equality of coefficients for correlated samples. Thus, group sparsity can be automatically discovered without prior knowledge. Utilizing this regularizer, Equation (1) can also be modified as the following optimization problem:

$$\min_{\boldsymbol{\alpha}_i} \frac{1}{2}\left\|\boldsymbol{x}_i - \boldsymbol{X}_{-i}\boldsymbol{\alpha}_i\right\|_2^2 + \lambda_1\|\boldsymbol{\alpha}_i\|_1 + \lambda_2\sum_{j<k}\max\{|\boldsymbol{\alpha}_i[j]|, |\boldsymbol{\alpha}_i[k]|\}, \tag{3}$$

where λ_1 and λ_2 are regularization parameters. In Fig. 3, we can see OSCAR uses a new penalty region that is octagonal in shape, which requires no initial information regarding the grouping structure.

Moreover, the real data are usually contaminated with sparse outlying entries or noise, thus we should extend the above problems to the noise case. Unlike the ℓ^1-graph introduced an identity matrix as a part of the dictionary to code the noise, we extend the auto-grouped sparse regularized optimization problem to a robust version as following:

$$\min_{\boldsymbol{\alpha}_i, \boldsymbol{e}_i} \frac{1}{2}\left\|\boldsymbol{x}_i - \boldsymbol{X}_{-i}\boldsymbol{\alpha}_i - \boldsymbol{e}_i\right\|_2^2 + \boldsymbol{\Omega}^g_{\lambda_1,\lambda_2}(\boldsymbol{\alpha}_i) + \lambda\|\boldsymbol{e}_i\|_1, \tag{4}$$

where $\boldsymbol{\Omega}^g_{\lambda_1,\lambda_2}(\cdot)$ denotes the auto-grouped sparse regularization, i.e., Elastic net or OSCAR. The above optimization problem is based on the standard least squares criterion and an ℓ^1-regularization term on noise term $\boldsymbol{e}_i$. Typically, if we set $\lambda_1 = \lambda$ and $\lambda_2 = 0$, Equation (4) will degenerate into Equation (1) used in Algorithm 1. However, since there is no closed-form solutions for this optimization problem, we proposed an efficient alternating procedure to obtain the optimal solution of Equation (4) with Accelerated Proximal Gradient (APG) method [24] and summarized the details in Algorithm 2.

Algorithm 2: The alternating optimization procedure for robust auto-grouped sparse regression.

Input : $\boldsymbol{x}_i \in \mathbb{R}^{m\times 1}$ and $\boldsymbol{X}_{-i} \in \mathbb{R}^{m\times(n-1)}$
Output: $\boldsymbol{\alpha}_i$, $\boldsymbol{e}_i$

1 Choose λ_1, λ_2, λ and group regularization $\boldsymbol{\Omega}^g_{\lambda_1,\lambda_2}$;
2 Set the Lipschitz constant $L = 2\sigma_{max}(\boldsymbol{X}_{-i}^\top\boldsymbol{X}_{-i})$;
3 Initialize $\boldsymbol{\alpha}_i^0 = \boldsymbol{0}$, $\boldsymbol{e}_i^0 = \boldsymbol{0}$;
4 **for** *t=1* **to** *maxiter* **do**
5 /* Compute $\boldsymbol{\alpha}_i^t$ when fix $\boldsymbol{e}_i^{t-1}$ by APG */
6 Initialize $k = 1$, $\omega_1 = 1$ and $\hat{\boldsymbol{\beta}}^1 = \boldsymbol{\beta}_0 = \boldsymbol{\alpha}_i^{t-1}$;
7 **repeat**
8 $\boldsymbol{\eta}^k \leftarrow \hat{\boldsymbol{\beta}}^k - \frac{1}{L}\boldsymbol{X}_{-i}^\top(\boldsymbol{X}_{-i}\hat{\boldsymbol{\beta}}^k - \boldsymbol{x}_i - \boldsymbol{e}_i^{t-1})$;
9 /* Proximal step */
10 $\boldsymbol{\beta}^k \leftarrow \arg\min_{\boldsymbol{\beta}}\{\Omega^g_{\lambda_1,\lambda_2}(\boldsymbol{\beta}) + \frac{L}{2}\|\boldsymbol{\beta} - \boldsymbol{\eta}^k\|_2^2\}$;
11 /* Update ω */
12 $\omega_{k+1} \leftarrow \frac{1+\sqrt{1+4\omega_k^2}}{2}$;
13 /* Update $\boldsymbol{\beta}$ */
14 $\hat{\boldsymbol{\beta}}^{k+1} \leftarrow \boldsymbol{\beta}^k + \frac{\omega_k - 1}{\omega_{k+1}}(\boldsymbol{\beta}^k - \boldsymbol{\beta}^{k-1})$;
15 $k \leftarrow k + 1$;
until *convergence*;
16 $\boldsymbol{\alpha}_i^t = \boldsymbol{\beta}^k$;
17 /* Compute $\boldsymbol{e}_i^t$ when fix $\boldsymbol{\alpha}_i^t$ */
18 $\boldsymbol{s} = \boldsymbol{x}_i - \boldsymbol{X}_{-i}\boldsymbol{\alpha}_i^t$;
19 /* Shrinkage operator */
20 $\boldsymbol{e}_i^t[j] = (\boldsymbol{s}[j] - \lambda)_+\text{sgn}(\boldsymbol{s}[j])$, $j \in \{1, 2, \cdots, m\}$;
21 /* Update */
22 $t \leftarrow t + 1$;
end

In the proposed Algorithm 2, we minimize the expression in Equation (4) alternately. First, we fix $\boldsymbol{e}_i$ and aim to find the best group sparse coefficient $\boldsymbol{\alpha}_i$. In this stage (i.e., Steps 5-15), we use APG to solve the problem efficiently, which has the convergence rate $\mathcal{O}(k^{-2})$ [24]. Meanwhile, by Propositions 1 and 2, the proximal step (i.e., Step 10) can be solved efficiently. Similarly, the second stage (i.e., Steps 17-20) fixes $\boldsymbol{\alpha}_i$ as known and seeks an update of $\boldsymbol{e}_i$ with a shrinkage operator. Finally, the iterative operation is terminated when the number of iterations is reached.

Time complexity. It is easy to see that computing $\boldsymbol{\eta}^k$ in Step 8 will take $\mathcal{O}(mn)$ time. For Steps 5-15, an ϵ-approximate solution of $\boldsymbol{\beta}^k$ can be obtained by APG in $\mathcal{O}(\frac{1}{\sqrt{\epsilon}})$ iterations [24]. In each iteration, Step 10 takes $\mathcal{O}(n)$ for Elastic net and $\mathcal{O}(n\log(n))$ for OSCAR [25]. For Steps 17-20, computing $\boldsymbol{e}_i^t$ only takes $\mathcal{O}(m)$ time. Hence, assuming the max iteration is $t = T$, Algorithm 2 takes a total of $\mathcal{O}(T(\frac{1}{\sqrt{\epsilon}}(m+1)n + m))$ time with Elastic net regularizer.

When we use the OSCAR regularizer instead, the total complexity is $\mathcal{O}(T(\frac{1}{\sqrt{\epsilon}}(m+\log n)n+m))$.

Proposition 1. *When* $\mathbf{\Omega}^{g}_{\lambda_1,\lambda_2}(\boldsymbol{\beta}) = \lambda_1\|\boldsymbol{\beta}\|_1 + \frac{\lambda_2}{2}\|\boldsymbol{\beta}\|_2^2$, *then the solution of the proximal step (Step 10 in Algorithm 2) is*

$$\boldsymbol{\beta}^*[i] = \frac{1}{1+\lambda_2}(|\boldsymbol{\eta}^k[i]-\lambda_1|)_+ sgn(\boldsymbol{\eta}^k[i]), i = \{1,2,\ldots,n-1\}. \tag{5}$$

Proposition 2. *Assuming that the indices* $i \in \{1,2,\ldots,n-1\}$ *have been permuted such that*

$$|\boldsymbol{\eta}^k[1]| \geq |\boldsymbol{\eta}^k[2]|, \ldots, \geq |\boldsymbol{\eta}^k[n-1]|, \tag{6}$$

thus if $\mathbf{\Omega}^{g}_{\lambda_1,\lambda_2}(\boldsymbol{\beta}) = \lambda_1\|\boldsymbol{\beta}\|_1 + \lambda_2\sum_{j<k}\max\{|\boldsymbol{\beta}[j]|, |\boldsymbol{\beta}[k]|\}$, *then for each group* $G_{s:t} = \{i \in \mathbb{Z} : s \leq i \leq t\}$, *the solution of the proximal step (Step 10 in Algorithm 2) is*[4]

$$z[s:t] = \frac{1}{t-s+1}\sum_{i\in G_{s:t}}\left(\boldsymbol{\eta}^k[i] - \frac{\lambda_1+\lambda_2(n-i-1)}{L}\right), \tag{7}$$

and

$$\boldsymbol{\beta}^*[i] = (z[i])_+ sgn(\boldsymbol{\eta}^k[i]), i \in G_{s:t}. \tag{8}$$

The result of Proposition 1 is well-known [15]. Also, we omit the proof of Proposition 2 which can be obtained from Theorem 1 in [25].

With these two auto-grouped regularization terms, we can discover the hidden data groups automatically and estimate the reconstruction sparse coefficient on a group-specific dictionary. Moreover, the learned data groups are consist of a small number of correlated samples. This means that the locality and sparsity properties can be preserved at the same time. Formally, we can define a set Λ_i to indicate the nonzero regression coefficients of $\boldsymbol{x}_i$, which are solved by the auto-grouped sparse representation. Indeed, $\Lambda_i = (\boldsymbol{\alpha}_i) = \{j : \boldsymbol{\alpha}_i[j] \neq 0\}$ but further emphasizes the datum indicated by Λ_i belongs to a neighborhood subspace and correlates with each other.

After inducing an alternate regularizer for promoting group sparsity, the construction process is formally stated in Algorithm 3.

Algorithm 3: GS-graph Construction

Input : Data matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{m\times n}$
Output: Affinity matrix $\boldsymbol{W} \in \mathbb{R}^{n\times n}$

1 Standardize the data matrix $\boldsymbol{X}$;
2 For each data point $\boldsymbol{x}_i$, find sparse coefficients $\boldsymbol{\alpha}_i^*$ by Algorithm 2 and normalize $\boldsymbol{\alpha}_i^*$;
3 Set affinity matrix $\boldsymbol{W}_{ij} = |\boldsymbol{\alpha}_i^*[j]|$ if $i > j$, $\boldsymbol{W}_{ij} = |\boldsymbol{\alpha}_i^*[j-1]|$ if $i < j$ and $\boldsymbol{W}_{ij} = 0$ if $i = j$;

Therefore, GS-graph in Algorithm 3 inherits the robustness and adaption of ℓ^1-graph, also achieves automatic group sparsity that is the lack of ℓ^1-graph. Conveniently, we term our two GS-graph as ℓ^1/ℓ^2-graph and ℓ^1/ℓ^∞-graph, respectively. Both construction methods can be summarized as follow:

ℓ^1/ℓ^2-graph (ℓ^1/ℓ^∞-graph). For $\forall i$, $i \sim j$, if $\boldsymbol{\alpha}_i^*[j] \neq 0$, $w_{ij} = |\boldsymbol{\alpha}_i^*[j]|$ when $i > j$ and $|\boldsymbol{\alpha}_i^*[j-1]|$ when $i < j$, $d(\boldsymbol{x}_i) = |\Lambda_i|$.

3.3 Kernelized Extension

The kernel trick is firstly introduced in support vector machine (SVM) to deal with classification problems that are not linearly separable in the origin feature space [26]. With the kernel, one can map the feature in the original space implicitly to a high or even infinite dimensional kernel feature space [27], [28]. A nonlinear problem can then be transferred into a liner problem in the kernel space since the nonlinear similarity of the original feature can be captured with the kernel. Thus, many manifold-based learning algorithms have been generalized to their kernelized versions, e.g., Kernel PCA [29], Kernel Discriminant Analysis [30], Kernel Locality Preserving Projections [4], etc. Recently, the kernel trick was introduced into sparse representation to building sparse models in the kernel feature space. For example, Kernel Sparse Representation (KSR) was proposed for improving the classification accuracy for object recognition [31]; A new classifier named Kernel Sparse Representation-based Classifier (KSRC) [32] was proposed as a nonlinear extension of Sparse Representation-based Classifier (SRC) [12]; A kernelized dictionary learning method was presented for learning the sparse and overcomplete signal representation in the kernel feature space [33].

In this section, we also extend our GS-graph to a kernelized version (named KGS-graph). In KGS-graph, the kernel sparse representation is utilized to derive the neighbor sample of a datum and its corresponding ingoing edge weights. Especially, we assume that the euclidean space $\mathbb{R}^m$ is mapped to a high or even infinite dimensional Hilbert space $\mathcal{H}$ through a nonlinear mapping function $\boldsymbol{\psi} : \mathbb{R}^m \to \mathcal{H}$. This Hilbert space is often known as a reproducing kernel Hilbert space (RKHS) corresponding to a Mercer kernel $k(\cdot,\cdot)$ [26]. Formally, a Mercer kernel $k(\cdot,\cdot)$ can be considered as an inner product $\langle\cdot,\cdot\rangle_{\mathcal{H}}$ in the kernel feature space $\mathcal{H}$. Therefore, given two samples $\boldsymbol{x}, \boldsymbol{z} \in \mathbb{R}^m$, we have $k(\boldsymbol{x},\boldsymbol{z}) = \langle\boldsymbol{\psi}(\boldsymbol{x}), \boldsymbol{\psi}(\boldsymbol{z})\rangle_{\mathcal{H}}$. Some commonly used kernels include the polynomial kernels $k(\boldsymbol{x},\boldsymbol{z}) = (\langle\boldsymbol{x},\boldsymbol{z}\rangle + c)^p$ and the Gaussian kernels $k(\boldsymbol{x},\boldsymbol{z}) = \exp -\frac{\|\boldsymbol{x}-\boldsymbol{z}\|_2^2}{c}$, where c and p are the parameters.

For the data points $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{m\times n}$, we define their images in the kernel feature space with $\boldsymbol{\Psi}(\boldsymbol{X}) = [\boldsymbol{\psi}(\boldsymbol{x}_1), \boldsymbol{\psi}(\boldsymbol{x}_2), \ldots, \boldsymbol{\psi}(\boldsymbol{x}_n)] \in \mathbb{R}^{K\times n}$, where K denotes the dimensionality of the kernel feature space $\mathcal{H}$. Usually, the dimensionality K is very large or infinite, thus it is necessary to construct a transformation matrix in $\mathcal{H}$ to project the images $\boldsymbol{\Psi}(\boldsymbol{X})$ form $\mathcal{H}$ into a reduced subspace. Following the notion in [32], we define the transformation matrix with $\boldsymbol{T} = [\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_d] \in \mathbb{R}^{K\times d}$ and each column of $\boldsymbol{t}_j$ is a linear combination of images in $\mathcal{H}$:

$$\boldsymbol{t}_j = \sum_{i=1}^{n}\boldsymbol{\gamma}_j[i]\boldsymbol{\psi}(\boldsymbol{x}_i) = \boldsymbol{\gamma}_j^\top\Psi(X). \tag{9}$$

4. Note that the element of z in each group G has a common value, and the groups can be generated by a series of merge operations with complexity $\mathcal{O}(n\log n)$. More details of the group merge algorithm can be found in Algorithm 2 of [25].

Let $\mathbf{\Gamma} = [\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \ldots, \boldsymbol{\gamma}_d] \in \mathbb{R}^{n\times d}$, then we have

$$T = \mathbf{\Psi}(X)\mathbf{\Gamma}. \tag{10}$$

The weight matrix $\mathbf{\Gamma}$ can be set by a random scheme [32].

Recalling Algorithm 3, group sparse coefficients are used to estimate the affinity matrix W for GS-graph construction. In Equation (4), the data $\boldsymbol{x}_i$ is reconstructed by the training data set X_{-i} in the origin feature space. Similarly, we can run the reconstruction in the kernel feature space $\mathcal{H}$ instead,

$$\min_{\boldsymbol{\alpha}_i, \boldsymbol{e}_i} \frac{1}{2}\left\|\boldsymbol{\psi}(\boldsymbol{x}_i) - \mathbf{\Psi}(\boldsymbol{X}_{-i})\boldsymbol{\alpha}_i - \boldsymbol{e}_i\right\|_2^2 + \mathbf{\Omega}^g_{\lambda_1,\lambda_2}(\boldsymbol{\alpha}_i) + \lambda\|\boldsymbol{e}_i\|_1, \tag{11}$$

where $\mathbf{\Psi}(X_{-i}) = [\mathbf{\Psi}(X) \setminus \boldsymbol{\psi}(\boldsymbol{x}_i)] \in \mathbb{R}^{K\times(n-1)}$. Moreover, integrating the transform matrix $\boldsymbol{T}$ into Equation (11), Equation (11) can be modified as

$$\min_{\boldsymbol{\alpha}_i, \boldsymbol{e}_i} \left\|\mathbf{\Gamma}^\top \boldsymbol{k}(\cdot, \boldsymbol{x}_i) - \mathbf{\Gamma}^\top \boldsymbol{K}_{-i}\boldsymbol{\alpha}_i - \boldsymbol{e}_i\right\|_2^2 + \mathbf{\Omega}^g_{\lambda_1,\lambda_2}(\boldsymbol{\alpha}_i) + \lambda\|\boldsymbol{e}_i\|_1, \tag{12}$$

where $\boldsymbol{k}(\cdot, \boldsymbol{x}) = [k(\boldsymbol{x}_1, \boldsymbol{x}), k(\boldsymbol{x}_2, \boldsymbol{x}), \ldots, k(\boldsymbol{x}_n, \boldsymbol{x})]^\top$, and $\boldsymbol{K}$ is the Gram matrix with $\boldsymbol{K}_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, so $\boldsymbol{K}_{-i} = [\boldsymbol{K} \setminus \boldsymbol{k}(\cdot, \boldsymbol{x}_i)] \in \mathbb{R}^{n\times(n-1)}$. Let $\tilde{x}_i = \mathbf{\Gamma}^\top \boldsymbol{k}(\cdot, \boldsymbol{x}_i)$ and $\tilde{X}_{-i} = \mathbf{\Gamma}^\top \boldsymbol{K}_{-i}$, the optimization of Equation (12) is equivalent to solving the problem in Equation (4), which can be solved by Algorithm 2.

So far, we have presented an effective sparse model with auto-grouped sparse regularization in the kernel feature space $\mathcal{H}$. Based on the kernelized sparse model, our GS-graph can be generalized to a kernelized version with Algorithm 4. Compared with Algorithm 3, the new proposed kernelized method emphasizes the endogenous sparse reconstruction (i.e., Step 3 in Algorithm 3) in the kernel space. While, the computation of Algorithm 4 will be more efficient when we set a low projection dimension d, i.e., $d < m$. For convenience, we summarize the time complexities of Algorithms 3 and 4 in Table 1.

Algorithm 4: KGS-graph Construction

Input : Data matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{m\times n}$
Output: Affinity matrix $\boldsymbol{W} \in \mathbb{R}^{n\times n}$

1 Choose a Mercer kernel $k(\cdot,\cdot)$ and set the weight matrix $\Gamma \in \mathbb{R}^{n\times d}$ with random project scheme;
2 Standardize the data matrix $\mathbf{\Gamma}^\top \boldsymbol{K}$;
3 For each data point $\boldsymbol{x}_i$, find sparse coefficient $\boldsymbol{\alpha}_i^*$ by Algorithm 2 and normalize $\boldsymbol{\alpha}_i^*$;
4 Set affinity matrix $\boldsymbol{W}_{ij} = |\boldsymbol{\alpha}_i^*[j]|$ if $i > j$, $\boldsymbol{W}_{ij} = |\boldsymbol{\alpha}_i^*[j-1]|$ if $i < j$ and $\boldsymbol{W}_{ij} = 0$ if $i = j$.

3.4 Local Grouping Property

In our method, we argue that both sparsity and locality (i.e., grouping) property are important for the neighbourhood graph construction. Without loss of generality, for a set of data $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$, let $\boldsymbol{\alpha}_i^* \in \mathbb{R}^{n-1}$ denotes the sparse representation of the ith data $\boldsymbol{x}_i \in \mathbb{R}^{m\times 1}$, which is obtained by minimizing Equations (1), (4) or (11). Then, the corresponding points of the nonzero elements of $\boldsymbol{\alpha}_i^*$ are set to be the neighbours of $\boldsymbol{x}_i$ and $\boldsymbol{\alpha}_i^*[j]$ denote the contribution of the jth point to the representation of $\boldsymbol{x}_i$. Solving $\boldsymbol{\alpha}_i^*$ for each data point, we can forms a neighbourhood graph $\mathcal{G}_X$ on $\mathcal{X}$.

TABLE 1
The Time Complexity of the Proposed Methods

	GS-graph	KGS-graph
Elastic net	$\mathcal{O}(\frac{T}{\sqrt{\epsilon}}mn^2)$	$\mathcal{O}(\frac{T}{\sqrt{\epsilon}}dn^2)$
OSCAR	$\mathcal{O}(\frac{T}{\sqrt{\epsilon}}(m+\log n)n^2)$	$\mathcal{O}(\frac{T}{\sqrt{\epsilon}}(d+\log n)n^2)$

The sparsity constraint can make the graph $\mathcal{G}_X$ tends to a sparse graph, i.e. each node $\boldsymbol{x}_i$ in $\mathcal{G}_X$ has a small degree $d_{\mathcal{G}}(x_i)$. A small degree ensures that no connections between data from different classes and makes the affinity matrix of $\mathcal{G}_X$ to be block diagonal [34].

Beyond the sparsity property, we also need the elements in the neighbours of a node $\boldsymbol{x}_i$ are corrected or local grouping. To achieve the locality property, it is expected that the sparse representation $\boldsymbol{\alpha}_i$ satisfy the following assumption: for $\forall j, k \in \{1, 2, \ldots, n-1\}$, if $\boldsymbol{x}_j \to \boldsymbol{x}_k$, then $\boldsymbol{\alpha}_i^*[j] \to \boldsymbol{\alpha}_i^*[k]$. Typically, the ℓ^1-norm regularized sparse representation does not have the local grouping property, which has been explicated theoretically in [35]. However, from the following Propositions 3 and 4, we can show that the solutions of Equations (4) and (11) in our method have the local grouping property theoretically. Then, this proves that the proposed GS-graph and KGS-graph can simultaneously achieve both locality and sparsity in graph construction.

Proposition 3. *Given standardized data matrix $\boldsymbol{X} \in \mathbb{R}^{m\times n}$, a normalized kernel $k(\cdot,\cdot)$ and parameters $(\lambda_1, \lambda_2, \lambda)$. For each data $\boldsymbol{x}_i$, let $\boldsymbol{\alpha}_i^*$ be the optimal solution of Equations (4) or (11) with $\mathbf{\Omega}^g_{\lambda_1,\lambda_2}(\boldsymbol{\alpha}_i) = \lambda_1\|\boldsymbol{\alpha}_i\|_1 + \frac{\lambda_2}{2}\|\boldsymbol{\alpha}_i\|_2^2$. Then, if $\boldsymbol{x}_j \to \boldsymbol{x}_k$, $\boldsymbol{\alpha}_i^*[j] \to \boldsymbol{\alpha}_i^*[k]$.*

Proposition 4. *Given standardized data matrix $\boldsymbol{X} \in \mathbb{R}^{m\times n}$, a normalized kernel $k(\cdot,\cdot)$ and parameters $(\lambda_1, \lambda_2, \lambda)$. For each data $\boldsymbol{x}_i$, let $\boldsymbol{\alpha}_i^*$ be the optimal solution of Equation (4) or (11) with $\mathbf{\Omega}^g_{\lambda_1,\lambda_2}(\boldsymbol{\beta}) = \lambda_1\|\boldsymbol{\beta}\|_1 + \lambda_2\sum_{j<k}\max\{|\boldsymbol{\beta}[j]|, |\boldsymbol{\beta}[k]|\}$. Then, if $\boldsymbol{x}_j \to \boldsymbol{x}_k$, $\boldsymbol{\alpha}_i^*[j] \to \boldsymbol{\alpha}_i^*[k]$.*

The proof of Propositions 3 and 4 can be found in the supplemental material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2014.2312322.

3.5 Graph-Based Learning Algorithms

In this section, we will show how to integrate our GS-graph with the following graph-based learning algorithms for diverse tasks: data embedding, clustering, subspace learning and manifold regularized non-negative matrix factorization.

(1) Embedding via GS-graph. The Laplacian embedding algorithm [2] is a geometrically motivated spectral algorithm for efficient nonlinear dimensionality reduction or embedding. Laplacian embedding can preserve the local topology of original data in the embedded space through an informative graph (any graph or our graph). Also, the embedded result can be obtained by solving the eigen-problem of a graph Laplacian matrix. Typically, when using the

group sparse graph for spectral embedding, the affinity matrix is automatically constituted with reconstruction coefficients $\boldsymbol{\alpha}_i^*$. Moreover, the group sparse coefficients are important to maintain "local topology". The detailed algorithm based on GS-graph is listed in Algorithm 5.

Algorithm 5: Spectral Embedding via GS-graph

Input : Data matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{m \times n}$
Output: Embedding data matrix $\boldsymbol{Y} \in \mathbb{R}^{d \times n}$, $d \ll m$
1 Construct the graph via Algorithm 3 or Algorithm 4;
2 Symmetrize affinity matrix $\boldsymbol{W} = (\boldsymbol{W} + \boldsymbol{W}^\top)/2$;
3 Set Laplacian matrix $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$, $\boldsymbol{D}_{ii} = \sum_j \boldsymbol{W}_{ij}$;
4 Compute eigenvalues and eigenvectors for the generalized eigenvector problem:$\boldsymbol{L}\boldsymbol{y} = \sigma \boldsymbol{D}\boldsymbol{y}$;
5 Let $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n$ be the eigenvectors, sorted in increasing according to each eigenvalues. Then the embedding data point matrix is given by: $\boldsymbol{Y} = [\boldsymbol{y}_2, \ldots, \boldsymbol{y}_{d+1}]^\top$.

(2) Subspace Learning via GS-graph. Sparsity preserving projections [7] is a recently developed subspace learning algorithm, in which the learning process is essentially achieved by constructing ℓ^1-graph. Furthermore, in [9], ℓ^1-graph is also used for a subspace learning. Based on the same notion, in this section, we develop a subspace learning algorithm with our GS-graph.

Basically, the generic problem subspace learning is to find a transformation matrix $\boldsymbol{A} \in \mathbb{R}^{m \times d}$ that maps each point $\boldsymbol{x}_i \in \mathbb{R}^m$ to a low dimension represent $\boldsymbol{y}_i \in \mathbb{R}^d (d \ll m)$, where $\boldsymbol{y}_i = \boldsymbol{A}^\top \boldsymbol{x}_i$. The transformation matrix $\boldsymbol{A}$ can be obtained through the following objective function

$$\min \sum_i \left\| \boldsymbol{A}^\top \boldsymbol{x}_i - \sum_j \boldsymbol{W}_{ij} \boldsymbol{A}^\top \boldsymbol{x}_j \right\|_2^2. \tag{13}$$

This function can be solved as the generalized eigenvector problem [4], [5]. Now, if we use GS-graph to construct affinity $\boldsymbol{W}$, the new subspace learning can be summarized in Algorithm 6.

Algorithm 6: Subspace Learning via GS-graph

Input : Data matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{m \times n}$
Output: Transformation matrix $\boldsymbol{A} \in \mathbb{R}^{m \times d}$
1 Construct the graph via Algorithm 3 or Algorithm 4;
2 Symmetrize affinity matrix $\boldsymbol{W} = (\boldsymbol{W} + \boldsymbol{W}^\top)/2$;
3 Solve the generalized eigenvector problem:

$$\boldsymbol{X}\boldsymbol{M}\boldsymbol{X}^\top \boldsymbol{a} = \sigma \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{a}, \tag{14}$$

where $\boldsymbol{M} = (\boldsymbol{I} - \boldsymbol{W})^\top (\boldsymbol{I} - \boldsymbol{W})$; $\boldsymbol{I} = diag(1, \ldots, 1)$;
4 Let $\boldsymbol{a}_1, , \boldsymbol{a}_1, \ldots, \boldsymbol{a}_d$ be the eigenvectors, sorted according to each eigenvalues $\sigma_1 \leq, \ldots, \sigma_d$. Then the transformation $\boldsymbol{A}$ is given by $\boldsymbol{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_d] \in \mathbb{R}^{m \times d}$.

(3) Non-negative matrix factorization via GS-graph. Non-negative matrix factorization is a popular algorithm to learn a part of the data representation, e.g., faces and text documents [6], [36], [37], [38]. Recently, graph or manifold regularization has been incorporated into NMF, named Graph-regularized non-negative matrix factorization (GNMF) [6]. GNMF achieves the state-of-the-art performance since it builds a new parts-based representation space to take advantage of the underlying geometrical structure of the data space. In this section, following the notion of GNMF, we utilize our GS-graph as regularization to non-negative matrix factorization.

Considering the data point matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{m \times n}$, NMF aims to find two non-negative matrices $\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_r] \in \mathbb{R}^{m \times r}$, and $\boldsymbol{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n] \in \mathbb{R}^{r \times n}$ such that $\boldsymbol{X} \approx \boldsymbol{U}\boldsymbol{V}$. Usually hidden factor r is chosen to be smaller than n or m. Thus, a compressed approximation can be rewritten column by column as $\boldsymbol{x}_i = \boldsymbol{U}\boldsymbol{v}_i, i = 1, 2, \ldots, n$. Therefore, $\boldsymbol{U}$ can be regarded as containing a basis that is optimized for linear combination of the data in $\boldsymbol{X}$. In GNMF [6], by integrating the manifold regularization [8], GNMF minimizes the objective function as follows:

$$\min \left\| \boldsymbol{X} - \boldsymbol{U}\boldsymbol{V} \right\|_F^2 + \nu \mathrm{Tr}(\boldsymbol{V}\boldsymbol{L}\boldsymbol{V}^\top), \tag{15}$$

where $\| \cdot \|_F$ denotes the Frobenius norm; $\mathrm{Tr}(\cdot)$ is the trace of matrix and $\boldsymbol{L}$ is the graph Laplacian matrix. The detailed algorithm based on GS-graph is described in Algorithm 7.

Algorithm 7: Non-negative Matrix Factorization via GS-graph

Input : Data matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{m \times n}$, hidden factor r, and $r \ll n$, $r \ll m$
Output: Non-negative basis matrix $\boldsymbol{U} \in \mathbb{R}^{m \times r}$ and coefficient matrix $\boldsymbol{V} \in \mathbb{R}^{r \times n}$
1 Constructing the graph via Algorithm 3 or 4;
2 Symmetrize affinity matrix $\boldsymbol{W} = (\boldsymbol{W} + \boldsymbol{W}^\top)/2$;
3 Set Laplacian matrix $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$, $\boldsymbol{D}_{ii} = \sum_j \boldsymbol{W}_{ij}$;
4 Solve the regularized optimization problem:

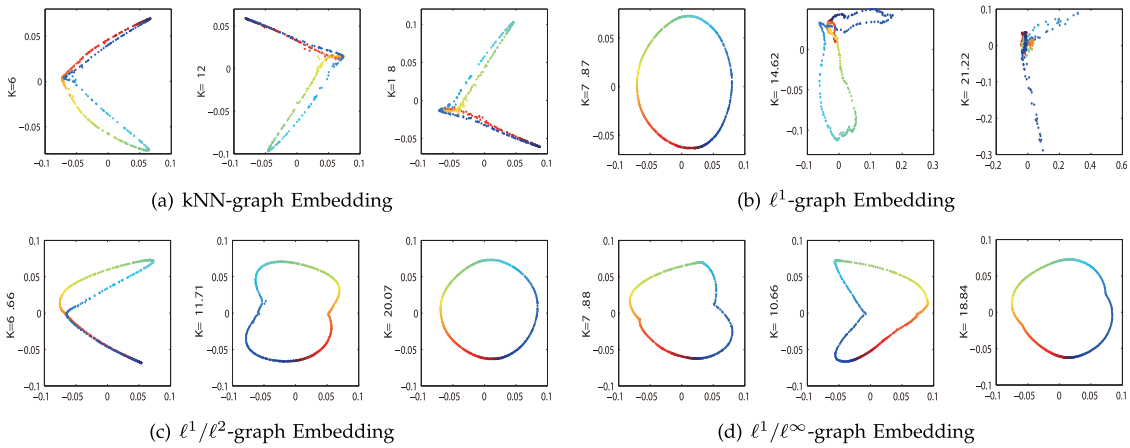
$$\min_{\boldsymbol{U}, \boldsymbol{V}} \| \boldsymbol{X} - \boldsymbol{U}\boldsymbol{V} \|_F^2 + \nu \mathrm{Tr}(\boldsymbol{V}\boldsymbol{L}\boldsymbol{V}^\top), \tag{16}$$

where ν is the regularized parameter;
5 Approximation of data $\boldsymbol{x}_i = \boldsymbol{U}\boldsymbol{v}_i, i = 1, 2, \ldots, n$.

4 Experiments

In this section, we evaluate the effectiveness of our proposed graph construction methods under different learning tasks, including data embedding, clustering, subspace learning and non-negative matrix factorization. All experiments are carried out by using Matlab/C++ on a PC with Inter 4-core 2.6GHz CPU and 12 GB RAM.

Parameter selection. In Algorithm 2, there are three essential parameters: λ_1, λ_2 and λ. The regularization parameter λ is used to tradeoff the penalty power for outliers and noises. In practice, we find $\lambda = 0.1$ is a good default setting, and we set the maximum iteration $T = 3$ (in Step 4) in all experiments. Moreover, for fair comparison with previous works, we follow the experimental setting in [9], [39] and search the parameters λ_1 and λ_2 from the candidate value set $\{10^{-1}, 10^{-2}, \ldots, 10^{-8}\}$ to achieve the best result. In Algorithm 4, we always choose the Gaussian kernel as the kernel metric, where $k(\boldsymbol{x}, \boldsymbol{z}) = \exp - \frac{\|\boldsymbol{x} - \boldsymbol{z}\|_2^2}{c}$ and the parameter c is set by the median value of $\|\boldsymbol{x}_i - \frac{1}{n}\sum_{i=1}^n \boldsymbol{x}_i\|_2^2$, $i = 1, 2, \ldots, n$.



(a) kNN-graph Embedding (b) ℓ^1-graph Embedding

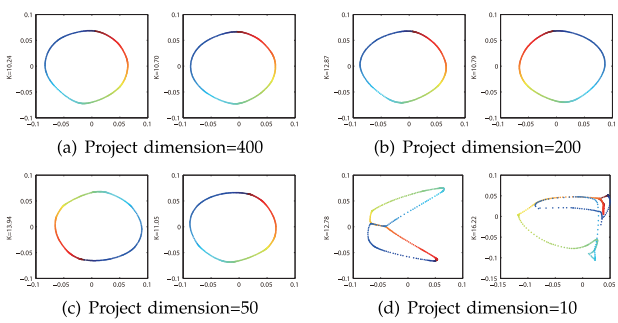
(c) ℓ^1/ℓ^2-graph Embedding (d) ℓ^1/ℓ^∞-graph Embedding

Fig. 4. 2D embedding of the teapot database obtained by: (a) kNN-graph (b) ℓ^1-graph (c) ℓ^1/ℓ^2-graph (d) ℓ^1/ℓ^∞-graph. For the visualization purpose, color coding is used to reveal how the data is embedding in two dimensions. The results are also computed for different choices of the numbers of nearest neighbors, K. K in (b), (c) and (d) are the average of the numbers of nonzero sparse coefficients.

4.1 Spectral Embedding

In this experiment, we compare our GS-graph based spectral embedding algorithm with Laplacian Eigenmaps and the ℓ^1-graph based spectral embedding algorithm. In the experiment, the teapot database is used, which contains 400 teapot color images (each of size $76 \times 101 \times 3$). The teapot was viewed in full 360 degrees of rotation. Theoretically, two-dimensional embedding of the database should be a circular, which can then reflect the underlying rotational degree of freedom [40]. In addition, noise has been added to each image to demonstrate that the proposed algorithm is insensitive to data noise. The results are shown in Fig. 4.

As one can see, embedding with kNN-graph does not succeed in either unravelling the manifold or recovering the two underlying degrees of freedom due to the influence of noise as shown in Fig. 4a. Although a reliable embedding can be obtained by ℓ^1-graph embedding with a small parameter, the results are unstable when adjusting the sparse coefficient to increase nonzero members as shown in Fig. 4b. In contract, a reliable and stable embedding manifold is obtained by our proposed methods (ℓ^1/ℓ^2-graph and ℓ^1/ℓ^∞-graph) with different numbers of nonzero sparse coefficients. Also, the two-dimensional embedding approximates a circular as shown in Figs. 4c and 4d.



(a) Project dimension=400 (b) Project dimension=200

(c) Project dimension=50 (d) Project dimension=10

Fig. 5. 2D embedding of the teapot database obtained by KGS-graph under different dimensions of random project matrix $\mathbf{\Gamma}$. For each subfigure (a)-(d), the left result is obtained by kernelized ℓ^1/ℓ^2-graph and the right result is obtained by kernelized ℓ^1/ℓ^∞-graph.

Moreover, we also evaluate the performance of our KGS-graph based spectral embedding algorithm. Especially, we use Algorithm 4 to construct an information graph in a special kernel feature space and then the embedding data matrix is obtained by Algorithm 5. In this experiment, the random linear project matrix $\mathbf{\Gamma} \in \mathbb{R}^{n \times d}$ in Algorithm 4 is set with different project dimension d. Fig. 5 shows how the result of spectral embedding via KGS-graph varies with the project dimension d. As we can see, our kernelized method is successful to learn the 2D embedding manifold and our method achieves consistently good performances with different d (i.e., d $=$ 400, 200, 50 or 10). When the project dimension is 10, the performance becomes not so good since the project dimension is too low to guarantee that the distances between data in high feature space are well-preserved under the random linear mapping $\mathbf{\Gamma}$. In addition, we also show the running times of each method in the Table 2. One can see that the running time of ℓ^1/ℓ^2-graph is smaller than ℓ^1/ℓ^∞-graph, which is consistent with our time complexity analysis in Table 1. For KGS-graph, the low running time can be obtained by setting a low project dimension. Thus, comparing to the GS-graph, it is possible to improve computation efficiency without losing the accuracy when inducing the kernel extension.

4.2 Spectral Clustering

In this section, we investigate the performance of our proposed method in spectral clustering. Spectral clustering is an unsupervised learning task, which will be used to compare our method with principal component analysis [14] and several different graph-based methods, e.g., LE-graph

TABLE 2
Running Time (Seconds) on the Teapot Database

	kNN-graph	ℓ^1-graph	ℓ^1/ℓ^2-graph	ℓ^1/ℓ^∞-graph
Times	1.27	119.20	83.23	190.76

	kernel ℓ^1/ℓ^2-graph	kernel ℓ^1/ℓ^∞-graph
Times	40.58, d=400	57.41, d=400
	24.40, d=200	31.80, d=200
	14.70, d=50	13.92, d=50
	8.20, d=10	7.95, d=10

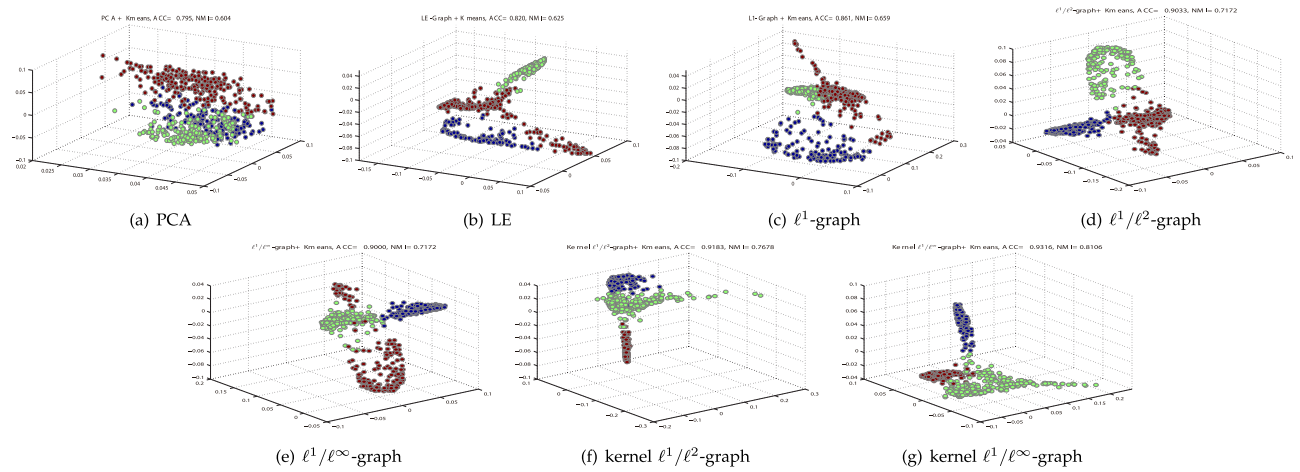


Fig. 6. Visualization of the clustering results on USPS. (a) PCA (b) LE (c) ℓ^1-graph (d) ℓ^1/ℓ^2-graph (e) ℓ^1/ℓ^∞-graph (f) kernel ℓ^1/ℓ^2-graph and (g) kernel ℓ^1/ℓ^∞-graph algorithm for three clusters (The handwritten digits 1, 2, and 3 in the USPS database). Three different colors of the points indicate the three different digits. Two compared metrics (ACC and NMI) are listed above the figures.

(Laplacian Eigenmaps) [2] and ℓ^1-graph based cluster method [9]. We choose K-means as our basic clustering algorithm. K-means is performed in the reduced feature space by PCA and other graph-oriented embedding methods. For visualization, the reduced dimension is set to be three. In our experiments, we use two widely used handwritten digit databases in clustering: USPS [41] and MINIST [42]. Following the method in [9], we use two standard metrics, the accuracy (ACC) and the normalized mutual information (NMI), to measure the clustering performance. Both ACC and NMI range from 0 to 1, while ACC reveals the clustering accuracy and NMI indicates whether the different clustering sets are identical (NMI $= 1$) or independent (NMI $= 0$). The details about these metrics can be found in [9] and [43].

For the USPS database, 200 randomly selected samples of each digit (i.e., 1, 2 and 3) from the database are used and the images are normalized to the size of 32×32 pixels. Fig. 6 shows the visualization of the clustering results. The images of digits (i.e., 1, 2 and 3) from the USPS database [41] are mapped into a 3-dimensional space and then clustered with K-means. As shown in the figure, compared with PCA, the LE and ℓ^1-graph got good results by preserving the embedded geometry structure. However, better results are obtained by ℓ^1/ℓ^2-graph, ℓ^1/ℓ^∞-graph and their kernelized versions (typically, we set $d = n$ in these experiments), where the data are much better separated by taking clustered sparsity into consideration in graph. Meantime, the proposed (kernel) group sparsity graph based spectral clustering algorithms outperform the other evaluated algorithms in the two qualitative metrics: ACC and NMI.

For the MINIST database, we construct two clustering tasks based on the first 3 and 7 subjects digit images. To reduce the computational cost, we also randomly select 200 samples from each subject and normalize the images to 32×32 pixels. Then Algorithm 5 is applied to these two tasks and the qualitative metrics are reported in Table 3. We can see that our GS-graph and KGS-graph outperform PCA, LE and ℓ^1-graph on all these two clustering tasks. In particular, KGS-graph performs slightly better than GS-graph, which can be attributed to the nonlinear extension of the data representation. Furthermore, to demonstrate the performance of the proposed methods, we also show the affinity matrixes obtained by our KGS-graph in Fig. 7. We can see that the affinity matrixes have an approximately block-diagonal structure that is similar to label arrangement.

4.3 Subspace Learning

In this experiment, we compare the (kernel) GS-graph based subspace learning algorithms with several representative unsupervised subspace learning techniques for face recognition [4], [5], [7], [14]. Two public face databases are selected for our experiments: the Extended Yale Face Database B (Extended Yale B) [44] and PIE face database [45].

The Extended Yale B database consists of a total of 38 individuals and each subject has around 64 frontal face

TABLE 3
Clustering Performance Comparisons on MINIST

	3 Subjects		7 Subjects	
Algorithms	**ACC**	**NMI**	**ACC**	**NMI**
PCA	0.5467	0.4099	0.4779	0.3991
LE	0.7850	0.5794	0.5179	0.5540
ℓ^1-graph	0.8017	0.6132	0.5957	0.6482
ℓ^1/ℓ^2-graph	0.8333	0.6289	0.6371	0.6539
ℓ^1/ℓ^∞-graph	0.8200	0.6323	0.6542	0.6111
kernel ℓ^1/ℓ^2-graph	**0.8483**	**0.7057**	**0.7093**	0.6328
kernel ℓ^1/ℓ^∞-graph	0.8317	0.6770	0.6914	**0.6806**

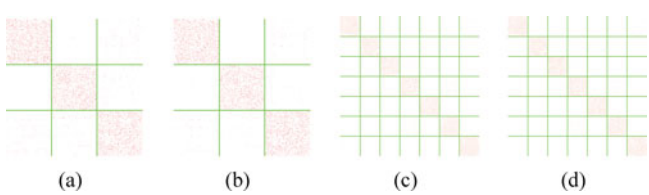


Fig. 7. Visualization of the KGS-graph affinity matrixes on MINIST database. (a) and (c) are the affinity matrices derived by kernel ℓ^1/ℓ^2-graph on 3 and 7 subjects respectively; (b) and (d) are the affinity matrices derived by kernel ℓ^1/ℓ^∞-graph.

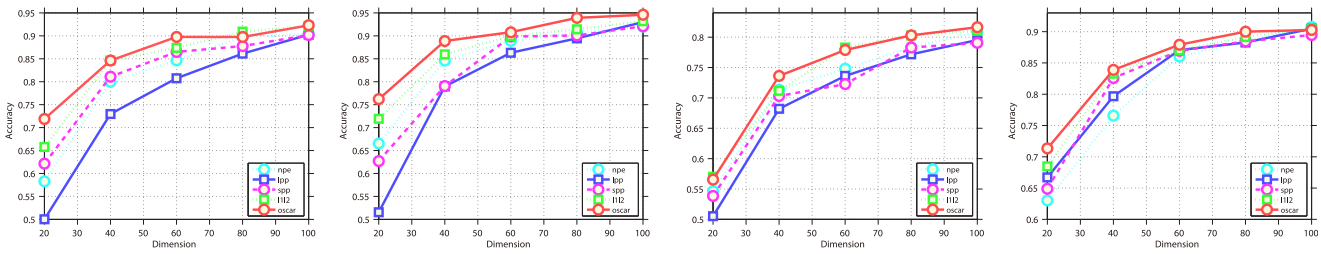


Fig. 8. The classification accuracies on the Extended Yale B and PIE database as a function of different subspace dimensions. (a) and (b) are the accuracies on the Extended Yale B with 30 and 50 trains respectively. (c) and (d) are the accuracies on the PIE with 15 and 30 trains, respectively.

TABLE 4
The Classification Accuracies on the Extended Yale B and PIE Databases

	Extended Yale B database						PIE database					
	30 trains			50 trains			15 trains			30 trains		
Algorithms	20d	60d	100d	20d	60d	100d	20d	60d	100d	20d	60d	100d
PCA	0.1282	0.4180	0.4501	0.1509	0.4682	0.5188	0.1126	0.2391	0.2641	0.1235	0.3518	0.3632
NPE	0.5829	0.8467	0.9043	0.6652	0.8889	0.9199	0.5465	0.7479	0.8091	0.6303	0.8603	0.9076
LPP	0.5002	0.8075	0.9026	0.5155	0.8630	0.9299	0.5053	0.7362	0.7953	0.6671	0.8706	0.9047
SPP	0.6214	0.8650	0.9015	0.6270	0.8983	0.9208	0.5385	0.7226	0.7906	0.6488	0.8694	0.8947
ℓ^1/ℓ^2-graph	0.6579	0.8737	0.9215	0.7192	0.8991	0.9319	**0.5703**	**0.7829**	0.8106	0.6847	0.8703	**0.9044**
ℓ^1/ℓ^∞-graph	**0.7189**	**0.8976**	**0.9227**	**0.7620**	**0.9078**	**0.9459**	0.5656	0.7788	**0.8165**	**0.7132**	**0.8791**	0.9026

images under various illuminations, while the PIE database contains 41,368 images of 68 subjects with different pose, illumination and expressions. Since the dimension of the original of face vectors in the two database is large, we downsize the sample images and each image is reduced and normalized to the size of 12×12 pixels. To evaluate the algorithmic performances on the databases, we randomly select different numbers (30 and 50 for Extended Yale B, 15 and 30 for PIE) of images from each individual for training and the rest are used for testing. Here, we use the classical nearest neighbor classifier for comparing the discriminating power from each subspace learning method, and the classification accuracies are used to measure the performance of different methods. The best results obtained in the different subspaces and the corresponding dimensionality (i.e., 20, 40, 60, 80 and 100) for each method are shown in Fig. 8 and Table 4.

In the test, the parameters for each method are tuned to achieve the best performance. In general, the performances of different methods vary with the numbers of dimensions and training sizes. However, from the classification accuracies shown in Fig. 8 and Table 4, one can see that the best results are obtained by the proposed GS-graph based subspace learning algorithms.

Furthermore, we examine the behavior of our kernel GS-graph for a subspace learning task. We illustrate the performance of our kernelized method under different project dimension d ($d = 500$ and 1,000). In the experiment, the training number per class is fixed to 30 for the Extended Yale B and 15 for PIE. Moreover, the subspace dimension is set to $\{20, 40, \ldots, 100\}$. In the test, we choose the Gaussian kernel and empirically tune the parameters for the best performance. As shown in Fig. 9, our kernelized methods perform well for different project dimensions with the Gaussian kernel. The classification accuracies are similar at different project dimension d and the higher dimension takes a slightly better accuracy.

4.4 Non-Negative Matrix Factorization

Non-negative matrix factorization is known as a powerful tool for feature extraction and can achieve good performance for clustering and classification tasks. In this section, we evaluate the proposed NMF via the GS-graph (GS-GNMF) algorithm in the image clustering task. To demonstrate that the clustering performance can be improved by our method, our method is compared with NMF [36] and group non-negative matrix factorization (GNMF) [6] on two databases: COIL20 [46] and COIL100 [47]. The COIL20 database contains 32×32 grey scale images of 20 objects viewed from different angles. Also, each object has 72 images from different viewpoints. Compared with COIL20 database, the COIL100 database is more challenging, which consists of 100 color objects (also 72 images per object). As a standard practice [6], we use two standard metrics: the accuracy and the normalized mutual information.

For the baseline method, we simply performed K-means in the original image space. In the test, GNMF has two parameters: the number of nearest neighbors k and the regularization parameter ν. Following the suggestion in [6], we set k to 5 and ν to 100. In GS-GNMF, we use the Elastic net as a group sparse regularization. Table 5 and Fig. 10 show the clustering results on COIL20 and COIL100 databases

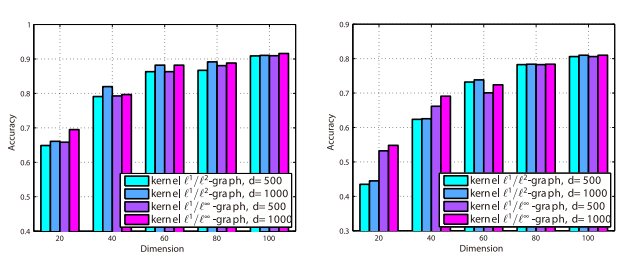
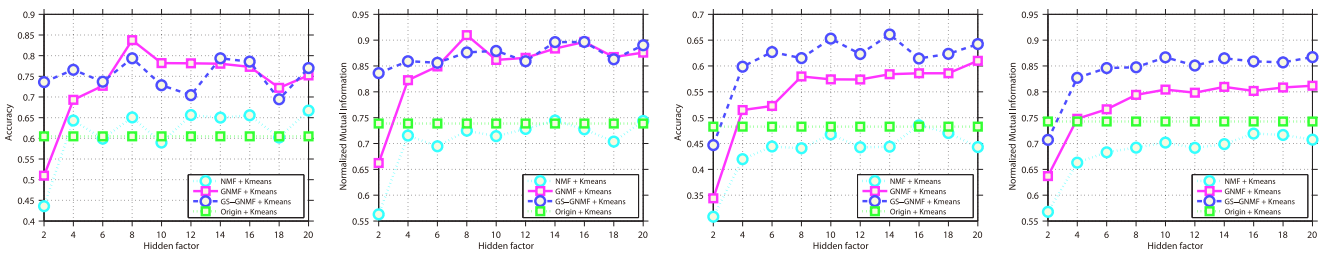


Fig. 9. The classification accuracies based on kernel GS-graph based subspace learning methods.

TABLE 5
Clustering Performance Comparisons on COIL20 and COIL100

	COIL20 database						COIL100 database					
	ACC			NMI			ACC			NMI		
r^*	NMF	GNMF	Our	NMF	GNMF	Our	NMF	GNMF	Our	NMF	GNMF	Our
2	0.4361	0.5097	**0.7361**	0.5629	0.6622	**0.8360**	0.3085	0.3440	**0.4468**	0.5679	0.6369	**0.7070**
4	0.6431	0.6931	**0.7659**	0.7153	0.8226	**0.8594**	0.4196	0.5147	**0.5988**	0.6631	0.7479	**0.8271**
6	0.5991	0.7270	**0.7368**	0.6947	0.8491	**0.8562**	0.4442	0.5226	**0.6271**	0.6829	0.7662	**0.8460**
8	0.6506	**0.8375**	0.7938	0.7247	**0.9097**	0.8763	0.4407	0.5797	**0.6151**	0.6921	0.7941	**0.8476**
10	0.5896	**0.7819**	0.7285	0.7143	0.8617	**0.8793**	0.4676	0.5742	**0.6529**	0.7019	0.8044	**0.8669**
12	0.6562	**0.7815**	0.7044	0.7281	**0.8658**	0.8593	0.4426	0.5739	**0.6231**	0.6916	0.7983	**0.8508**
14	0.6500	0.7805	**0.7936**	0.7445	0.8837	**0.8962**	0.4438	0.5840	**0.6608**	0.6986	0.8096	**0.8647**
16	0.6556	0.7729	**0.7854**	0.7274	**0.8966**	0.8965	0.4851	0.5859	**0.6146**	0.7193	0.8017	**0.8589**
18	0.6020	**0.7222**	0.6944	0.7037	**0.8674**	0.8630	0.4704	0.5860	**0.6233**	0.7166	0.8083	**0.8570**
20	0.6673	0.7522	**0.7701**	0.7436	0.8759	**0.8902**	0.4430	0.6100	**0.6422**	0.7075	0.8118	**0.8672**
Avg.	0.6105	0.7359	**0.7509**	0.7059	0.8495	**0.8712**	0.4366	0.5475	**0.6105**	0.6842	0.7779	**0.8393**

$^*r$ is the hidden factor



(a) ACC on COIL20 (b) NMI on COIL20 (c) ACC on COIL100 (d) NMI on COIL100

Fig. 10. Comparison of clustering accuracy and normalized mutual information of different NMF methods on COIL20 and COIL100 databases.

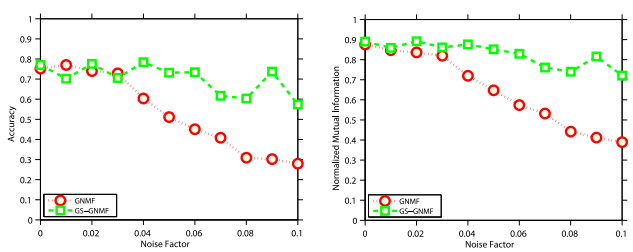
with different hidden factor r. One can see that both GS-GNMF and GNMF result in the best performance by inducing the graph structure. In particular, GS-GNMF achieves a slightly better performance than GNMF on COIL20. But for the larger database COIL100, GS-GNMF achieves significant improvements.

When we add noise to the images, the performance of GNMF decreased drastically with the increase of noise factor (typically, the noise factor is the model parameter to control the noise level), as shown in Fig. 11. This is because GNMF uses the k-nearest neighbor graph to capture the local geometric structure, which is sensitive to noise. However, the performance of GS-GNMF decreases slightly when σ increases as shown in Fig. 11. This demonstrates GS-GNMF is not so sensitive to noise.

Moreover, we also test the performance of NMF with KGS-graph on the COIL20. We use a Gaussian kernel and hidden factor $r = 20$. In order to verify the effectiveness of our method, the dimensionality of the approximate kernel feature space in Algorithm 4 is set to be in the range from 100 to 1,000. The accuracy and the normalized mutual information under different project dimensions are shown in Fig. 12. From the results, one can see that our kernelized method achieves good performances with different approximate project dimensions. However, when the project dimension becomes high, both the ACC and NMI increase slightly. This means the approximate feature with high dimensionality contains more discriminative information.

5 Conclusion

In this paper, we present a new information graph: Group Sparse graph, which is based on ℓ^1-graph to integrate the properties of sparsity and locality simultaneously. Firstly, we illustrate the limitation of ℓ^1-norm in ℓ^1-graph to construct an information graph, which causes ℓ^1-graph to be strong in sparsity but weak in locality. Thus, we introduce and extend two sparse regularization terms for our graph construction, i.e., Elastic net and OSCAR, which have an auto-grouping effect. With these two regularization terms, the solved nonzero reconstruction coefficients for building the graph have a clustered sparsity form, which makes our proposed graph have both sparsity and locality properties in graph construction. Moreover, we extend the proposed graph to a kernelized version (KGS-graph) with the kernel trick. In the KGS-graph algorithm, two auto-grouping



(a) ACC (b) NMI

Fig. 11. Clustering performance comparisons between GNMF and GS-GNMF on COIL20 under different noise levels.

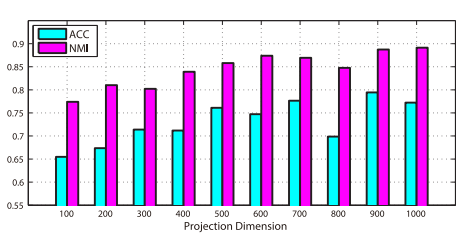


Fig. 12. Accuracy and normalized mutual information. Clustering performance of NMF via KGS-graph under different project dimensions.

sparse regularization terms (Elastic net and OSCAR) are modified to estimate the reconstruction coefficients in a high dimensional kernel feature space. Thus, with KGS-graph, more discriminative coefficients can be learned to improve the graph construction for the data classification and clustering. Finally, we integrate the group sparse graph with various graph-oriented learning algorithms: spectral embedding, spectral clustering, subspace learning and non-negative matrix factorization. The experimental results on each task and data sets show that the proposed GS-graph method outperforms traditional graph construction methods [2], [4], [5], [14] and the ℓ^1-graph method [7], [9]. Furthermore, since graph is widely used in computer vision and machine learning, our technique can be applied in other latest graph-oriented learning algorithms, e.g., graph regularized sparse coding [48] and graph-based ranking for image retrieval [49].

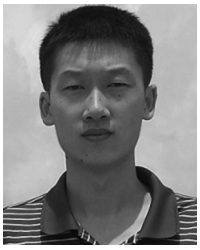
Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable and constructive comments on improving the paper. This research is supported by the National Natural Science Foundation of China (No. 61075043, No. 91220301, No. 61229301), the Program for Changjiang Scholars and Innovative Research Team in University (PCSIRT) of the Ministry of Education, China, under grant IRT13059, and the National 973 Program of China under grant 2013CB329604.

References

[1] J. Tenenbaum, V. Silva, and J. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.
[2] M. Belkin and M. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Comput.*, vol. 15, no. 6, pp. 1373–1396, 2003.
[3] R. Sam and S. Lawrence, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.
[4] X. He and P. Niyogi, "Locality preserving projections," in *Proc. Adv. Neural Inf. Process. Syst. 16*, 2003, pp. 9–16.
[5] X. He, D. Cai, S. Yan, and H. Zhang, "Neighborhood preserving embedding," in *Proc. Int. Conf. Comput. Vis.*, 2005, pp. 17–21.
[6] D. Cai, X. He, J. Han, and T. S. Huang, "Graph regularized nonnegative matrix factorization for data representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1548–1560, Aug. 2011.
[7] L. Qiao, S. Chen, and X. Tan, "Sparsity preserving projections with applications for face recognition," *Pattern Recog.*, vol. 43, no. 1, pp. 331–341, 2010.
[8] B. Milhail, N. Partha, and S. Vikas, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *J. Mach. Learn. Res.*, vol. 7, no. 48, pp. 2399–2434, 2006.
[9] B. Cheng, J. Yang, S. Yan, Y. Fu, and T. Huang, "Learning with ℓ^1-graph for image analysis," *IEEE Trans. Image Process.*, vol. 19, no. 4, pp. 858–866, Apr. 2010.
[10] S. Yan, D. Xu, B. Zhang, H. Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: A general framework for dimensionality reduction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 1, pp. 40–51, Jan. 2007.
[11] T. Robert, "Regression shrinkage and selection via the lasso," *J. Roy. Statistical Soc. Ser. B*, vol. 58, no. 1, pp. 267–288, 1996.
[12] J. Wright, A. Yang, A. Ganesh, S. Sastry, and M. Yi, "Robust face recognition via sparse representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 210–227, Feb. 2009.
[13] R. Timofte and L. Van Gool, "Sparse representation based projections," in *Proc. 22nd Brit. Mach. Vis. Conf.*, 2011, pp. 61.1–61.12.
[14] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *J. Educ. Psychol.*, vol. 29, no. 1, pp. 40–51, 1993.
[15] H. Zou and H. Hastie, "Regression and variable selection via the elastic net," *J. Roy. Statistical Soc. Ser. B*, vol. 67, no. 2, pp. 301–320, 2005.
[16] H. Bondell and B. Reich, "Simultaneous regression shrinkage, variable selection and supervised clustering of predictors with oscar," *Biometrics*, vol. 64, no. 1, pp. 115–123, 2008.
[17] Y. Fang, R. Wang, and B. Dai, "Graph-oriented learning via automatic group sparsity for data analysis," in *Proc. 12th IEEE Int. Conf. Data Mining*, 2012, pp. 251–259.
[18] D. Reinhard, *Graph Theory*. Heidelberg, Germany: Springer-Verlag, 2010.
[19] F. Coleman and Thmas J. More, Jorge, "Estimation of sparse jacobian matrices and graph coloring problems," *SIAM J. Numer. Anal.*, vol. 20, no. 1, pp. 187–209, 1983.
[20] S. Lang, *Introduction to Differentiable Manifolds*. New York, NY, USA: Springer-Verlag, 2002.
[21] A. Majumdar and R. Ward, "Classification via group sparsity promoting regularization," in *Proc. 32th Int. Conf. Acoust., Speech Signal Process*, 2009, pp. 861–864.
[22] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *J. Roy. Statistical Soc. Ser. B*, vol. 68, no. 1, pp. 49–67, 2006.
[23] M. Stojnic, F. Parvaresh, and B. Hassibi, "On the reconstruction of block-sparse signals with an optimal number of measurements," *IEEE Trans. Signal Process.*, vol. 57, no. 8, pp. 3075–3085, Aug. 2009.
[24] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imaging Science*, vol. 2, no. 1, pp. 183–202, 2009.
[25] W. Zhong and J. Kwok, "Efficient sparse modeling with automatic feature grouping," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 9, pp. 1436–1447, Sep. 2012.
[26] B. Scholkopf and A. J. Smola, *Learn. with Kernels: Support Vector Machine, Regularization, Optimization, and Beyond*. New York, NY, USA: The MIT Press, 2001.
[27] T. Hormann, B. Scholkopf, and A. J. Smola, "Kernel methods in machine learning," *The Ann. Statist.*, vol. 36, no. 3, pp. 1171–1220, 2008.
[28] M. E. Abbasnejad, D. Ramachandram, and R. Mandava, "A survey of the state of the art in learning the kernels," *Knowl. Inf. Syst.*, vol. 31, no. 2, pp. 193–221, 2012.
[29] B. Scholkopf, A. Smola, and K. R. Muller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, no. 1, pp. 1299–1319, 1998.
[30] G. Baudat and F. Anouar, "Generalized discriminant analysis using a kernel approach," *Neural Comput.*, vol. 12, no. 10, pp. 2385–2398, 2000.
[31] S. Gao, I. W. -H. Tsang, and L. -T. Chia, "Sparse representation with kernels," *IEEE Trans. Image Process.*, vol. 22, no. 2, pp. 423–434, Feb. 2013.
[32] L. Zhang, W. Zhou, P. Chang, J. Liu, Z. Yan, T. Wang, and F. Li, "Kernel sparse representation-based classifier," *IEEE Trans. Signal Process.*, vol. 60, no. 4, pp. 1684–1695, Apr. 2012.
[33] H. V. Nguyen, V. M. Patel, N. M. Nasrabadi, and R. Chellappa, "Kernel dictionary learning," in *Proc. 32nd Int. Conf. Acoust., Speech Signal Process.*, 2012, pp. 861–864.
[34] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2765–2781, Nov. 2013.
[35] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Ann. Statist.*, vol. 32, no. 2, pp. 407–499, 2004.
[36] L.Daniel D and H. S. Seung, "Learning the parts of objects by non-negative matrix facrization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.
[37] H. Ma, W. Zhao, and Z. Shi, "A nonnegative matrix factorization framework for semi-supervised document clustering with dual constraints," *Knowl. Inf. Syst.*, vol. 36, no. 3, pp. 629–651, 2013.
[38] Z.-Y. Zhang, T. Li, and C. Ding, "Non-negative tri-factor tensor decomposition with applications," *Knowl. Inf. Syst.*, vol. 34, no. 2, pp. 243–265, 2013.
[39] C.-Y. Lu, H. Min, Z.-Q. Zhao, L. Zhu, D.-S. Huang, and S. Yan, "Robust and efficient subspace segmentation via least squares regression," in *Proc. 12th Eur. Conf. Comput. Vis.*, 2012, pp. 347–360.

[40] Weinberger, Saul, "An introduction to nonlinear dimensionality reduction by maximum variance unfolding," in *Proc. Nat. Conf. Artif. Intell.*, 2006, pp. 1683–1686.
[41] J. Hull, "A database for handwritten text recognition research," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 5, pp. 550–554, May 1994.
[42] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proc. IEEE.*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[43] X. Zheng, D. Cai, X. He, W. Ma, and X. Lin, "Locality preserving clustering for image database," in *Proc. ACM Int. Conf. Multimedia*, 2004, pp. 885–891.
[44] A. Georghiades, P. Belhumeur, and D. Kriegman, "From few to many: Illumination cone models for face recognition under variable lighting and pose," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 6, pp. 643–660, Jun. 2001.
[45] T. Sim, S. Baker, and M. Bsat, "The CMU pose, illumination, and expression database," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 12, pp. 1615–1618, Dec. 2003.
[46] S. Nene, S. Nayar, and H. Murase, "Columbia object image library (COIL-20)," Tech. Rep. CUCS-005-96, 1996.
[47] S. Nene, S. Nayar, and H. Murase, "Columbia object image library (COIL-100)," Rep. CUCS-006-96, 1996.
[48] M. Zheng, J. Bu, C. Chen, C. Wang, L. Zhang, G. Qiu, and D. Cai, "Graph regularized sparse coding for image representation," *IEEE Trans. Image Process.*, vol. 20, no. 5, pp. 1327–1336, May 2011.
[49] X. Bin, B. Jiajun, C. Chun, C. Deng, H. Xiaofei, L. Wei, and L. Jiebo, "Efficient manifold ranking for image retrieval," in *Proc. 34th Annu. Int. ACM SIGIR Conf.*, 2011, pp. 885–891.



Yuqiang Fang received the master's degree in control science and engineering, in 2010, and is currently working toward the PhD degree in pattern recognition and intelligent systems from the National University of Defense Technology (NUDT), Hunan, P. R. China. His research interests include data mining, computer vision, machine learning and autonomous vehicle.

Ruili Wang received the PhD degree in computer science from Dublin City University. He is currently a senior lecturer at Massey University. His research interests include data mining, machine learning, intelligent systems and speech processing. He has received one of the most prestigious research grants in New Zealand, Marsden Fund. He is an associate editor and member of editorial boards of five international journals.

Bin Dai received the PhD degree in control science and engineering from the National University of Defense Technology (NUDT), Hunan, P. R. China, in 1998. He is currently a professor in the College of Mechatronic Engineering and Automation at NUDT. His current research interests include pattern recognition, data mining and autonomous vehicle.

Xindong Wu received the PhD degree in artificial intelligence from the University of Edinburgh, United Kingdom. He is a Yangtze River scholar in the School of Computer Science and Information Engineering, Hefei University of Technology, China, and a professor of Computer Science at the University of Vermont. His research interests include data mining and knowledge-based systems. He is the Steering Committee chair of the IEEE International Conference on Data Mining, and the editor-in-chair of *Knowledge and Information Systems*. He is a fellow of the IEEE and AAAS.