Neurocomputing 120 (2013) 441–452

Contents lists available at ScienceDirect

Neurocomputing

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

# $L_1$-graph construction using structured sparsity

Guangyao Zhou, Zhiwu Lu, Yuxin Peng *

*Institute of Computer Science and Technology, Peking University, Beijing 100871, China*

ARTICLE INFO

*Article history:*
Received 4 June 2012
Received in revised form
2 January 2013
Accepted 24 March 2013
Communicated by Xianbin Cao
Available online 12 June 2013

*Keywords:*
Graph construction
Sparse representation
$L_1$-graph
Structured sparsity
Spectral clustering

ABSTRACT

As a powerful model to represent the data, graph has been widely applied to many machine learning tasks. More notably, to address the problems associated with the traditional graph construction methods, sparse representation has been successfully used for graph construction, and one typical work is $L_1$-graph. However, since $L_1$-graph often establishes only part of all the valuable connections between different data points due to its tendency to ignore the intrinsic structure hidden among the data, it fails to exploit such important information for the subsequent machine learning. Besides, the high computational costs of $L_1$-graph prevent it from being applied to large scale high-dimensional datasets. In this paper, we construct a new graph, called the $k$-nearest neighbor ($k$-NN) fused Lasso graph, which is different from the traditional $L_1$-graph because of its successful incorporation of the structured sparsity into the graph construction process and its applicability to large complex datasets. More concretely, to induce the structured sparsity, a novel regularization term is defined and reformulated into a matrix form to fit in the sparse representation step of $L_1$-graph construction, and the $k$-NN method and kernel method are employed to deal with large complex datasets. Experimental results on several complex image datasets demonstrate the promising performance of our $k$-NN fused Lasso graph and also its advantage over the traditional $L_1$-graph in the task of spectral clustering.



## 1. Introduction

Since graph is a powerful model to represent the data, it has served as foundation for lots of machine learning problems, such as spectral clustering [1,2], semi-supervised learning [3,4], dimension reduction [5] and so on. Although many graph-based methods have been developed for different machine learning tasks, graph construction still receives relatively little attention as pointed out in [6,7]. In the literature, there exist two commonly used strategies for graph construction, namely $k$-nearest neighbor ($k$-NN) and $\epsilon$−ball methods. Although these methods are easy both to understand and to implement, they suffer from inherent limitations, e.g. data dependency and sensitivity to noise.

Recently, to address these problems, sparse representation [8] has been successfully used for graph construction, among which one typical work is $L_1$-graph [9,10]. The success of $L_1$-graph lies in the sparse representation step, in which it seeks a sparse linear reconstruction of each data point with the other data points by exploiting the sparse property of the Lasso penalty [11]. This is, in fact, a new way that is fundamentally different from the traditional ones (like Euclidean distance, cosine distance, etc.) to measure the similarity between different data points. By inducing sparsity in the linear reconstruction process, it identifies the most relevant data points as well as their estimated similarity to the reconstructed data point, and by doing so gets a graph that proves effective in the subsequent graph-based machine learning tasks.

However, two interesting comments from previous works on the Lasso method and the $L_1$-graph attract our attention:

(1) As reported in [12], when faced with a group of highly correlated variables, Lasso method tends to randomly choose one of them.
(2) In [10], the authors stated that "for certain extreme cases, e.g. if we simply duplicate each sample and generate another new dataset of double size, $L_1$-graph may only connect these duplicated pairs".

With some simple mathematical derivations, as shown in Section 2, we can see the similarity between the sparse representation step of $L_1$-graph construction and the Lasso method. As a result, if we think of the data points and the similarity between data points in sparse representation step as variables and correlation between variables in Lasso method respectively, the first comment indeed suggests that the sparse representation does not connect all the data points that need to be connected. In the situation mentioned in the second comment, the similarity between the reconstructed data point and its duplicate (measured by the sparse representation method of $L_1$-graph) dominates others, which also makes the sparse representation step ignore many other valuable connections.

In addition, advances in technology have made large scale high-dimensional datasets common in many scientific disciplines,

* Corresponding author. Tel.: +86 10 82529699; fax: +86 10 82529207.
*E-mail addresses:* tczhouguangyao@gmail.com (G. Zhou), luzhiwu@pku.edu.cn (Z. Lu), pengyuxin@pku.edu.cn (Y. Peng).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2013.03.045

yet the construction process of $L_1$-graph, in which the computational costs become unbearable because of a huge matrix (details will be given in Section 2) constructed when dealing with these datasets, prevents it from being further applied to problems related to such large complex datasets.

Our work mainly aims to overcome these shortcomings of $L_1$-graph. To avoid $L_1$-graph's failure to establish all valuable connections between different data points, we seek to incorporate structured sparsity into the $L_1$-graph construction process. The main idea is to exploit the local structure across the dataset by making the reconstruction coefficients of every data point and its nearest neighbors also close to each other in value in the linear reconstruction process of the sparse representation step. To achieve this, we propose a novel regularization term, which makes use of the information provided by traditional ways of measuring similarity between data points, for the sparse representation step of $L_1$-graph construction to induce structured sparsity and reformulate it in matrix form to fit in our new graph (which we call $k$-NN fused Lasso graph) construction process. And in order to deal with large scale high-dimensional datasets, we employ the $k$-NN method and kernel method in our new graph construction process. To be more specific, we reconstruct each data point and construct the corresponding new regularization term, both with only its $k$ nearest neighbors to handle large scale datasets. And when solving the linear reconstruction problem, we use the kernel matrix instead of the original data vectors to handle high-dimensional datasets. The effectiveness of $k$-NN fused Lasso graph is verified by the experimental results on several large complex image datasets in the task of spectral clustering. Specifically, to gain a first impression of its effectiveness, the similarity (i.e. weight) matrix of our new graph on the doubled soybean dataset (the soybean dataset, which contains 47 35-dimensional instances, can be downloaded from the UCI Machine Learning Repository [20], and the doubled soybean dataset is generated by making an exact duplicate of each data point in the original dataset) are illustrated on Fig. 1(b). Comparing it with Fig. 1(a), we can easily see the tremendous advantage of our new graph over the $L_1$-graph.

Our main contribution is the development of the new $k$-NN fused Lasso graph construction method. To be more specific, our contributions can be summarized as follows:

(1) We proposed a novel regularization term to induce structured sparsity.
(2) We designed a reformulation strategy to incorporate the new regularization term into the graph construction process.
(3) We successfully employed the $k$-NN method and kernel method to make our graph construction method applicable to large scale high-dimensional datasets.

The idea of linearly reconstructing a given data point by its neighbors is also used in some other works, e.g. the locally linear embedding [22] method for dimension reduction. However, unlike our method, these works did not pay much attention to the reconstruction process itself. In [23], the authors proposed a unifying framework for dimension reduction called patch alignment. In our graph construction process, by using the $k$-NN method, we also construct a patch for each data point, and conducting the sparse representation step is similar to the part optimization in [23]. By unlike [23], we do not have a whole alignment step. We run the sparse representation step for each of the patches, and unifying them in the end to get the similarity matrix by symmetrizing the original similarity matrix constructed by the sparse representation steps. The idea of exploring the dataset structure in a pairwise manner is also present in some previous works, e.g. the max-min distance analysis [24]. But in [24], the authors used the pairwise distance between different classes, while our method focuses on the pairwise distance of the reconstruction coefficients of different data points. Also, our method is unsupervised in nature. We do not need such prior information as class labels, which makes our method applicable to many unsupervised or semi-supervised problems, and distinguishes our work from previous works like [24,14], as well as some other works, like the Group Sparse MahNMF in [25]. Like the elastic net [12], our new regularization term also has certain grouping effect. But we promote such grouping effect in a pairwise manner with the $L_1$ norm, which makes our method performs differently from the elastic net [12] as well as some other elastic net based works, such as Elastic Net Inducing MahNMF [25] and Manifold Elastic Net [26]. To the best of our knowledge, we have made the first attempt to incorporate the structured sparsity into the $L_1$-graph construction process, and the fact that our new $k$-NN fused Lasso graph outperforms the traditional $k$-NN graph and $L_1$-graph (see later experimental results in Section 6) when applied to spectral clustering on large complex image datasets demonstrates the great value of the structured sparsity information we utilize in our new method.

The rest of the paper will be organized as follows. In Section 2, we briefly review the $L_1$-graph construction method. In Section 3, we describe in detail how we overcome the shortcomings of

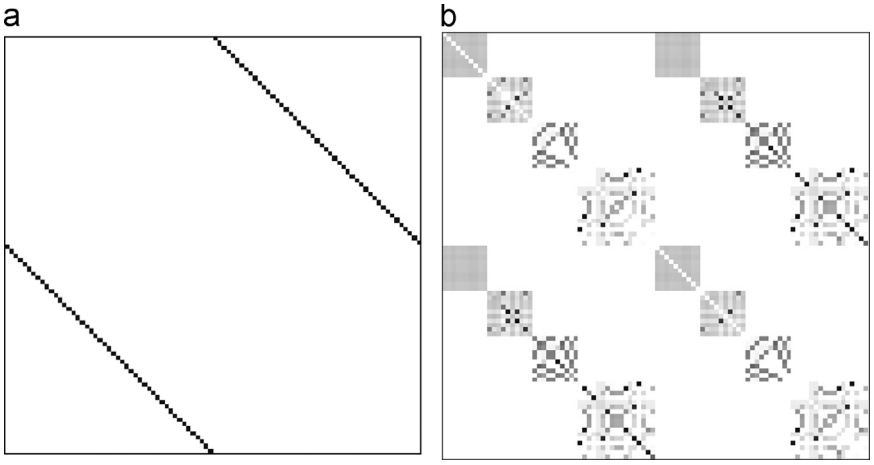


**Fig. 1.** Comparison between the similarity (i.e. weight) matrices of the $L_1$-graph and our $k$-NN fused Lasso graph on the doubled soybean dataset. For illustration purpose, the first and second half of the new dataset are identical copies of the original dataset, and each copy of the original dataset is rearranged such that data points within a class appear consecutively. More notably, the darker is a pixel, the larger is the similarity. (a) $L_1$-graph and (b) our graph.

$L_1$-graph by the new regularization term, $k$-NN method and kernel method, and summarize the $L_1$-graph construction process used in this paper, which also makes use of the $k$-NN method and kernel method to deal with large complex image datasets. In Section 4, we summarize the construction process of the new $k$-NN fused Lasso graph. In Section 5, we compare the computational complexity of the different graph construction methods. Section 6 provides the experimental results to verify the effectiveness of our new $k$-NN fused Lasso graph when applied to spectral clustering, and Section 7 gives conclusions.

## 2. $L_1$-graph construction

In this section, we give a brief review of the closely connected $L_1$-graph construction method [10]. Suppose we have a set of data points $a_1, a_2, \ldots, a_n$, which are represented in the form of column vectors ($a_i \in \mathbb{R}^m$). Our goal is to construct an $L_1$-graph based on these data points. Motivated by the limitations of the traditional graph construction methods as mentioned above, the $L_1$-graph construction method seeks to determine the neighborhood and the edge weight simultaneously. The corresponding algorithm [10] is outlined as follows:

(1) *Inputs*: The set of data points $a_1, a_2, \ldots, a_n$, where $a_i \in \mathbb{R}^m$.
(2) *Sparse representation*: For each data point $a_i$, Let

$$B^i = [a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n] \tag{1}$$

We solve the optimization problem

$$\min_{x^i} \|x^i\|_1, \quad \text{s.t. } A^i x^i = a_i \tag{2}$$

where $A^i = [B^i, I] \in \mathbb{R}^{m \times (n+m-1)}$, $x^i \in \mathbb{R}^{n+m-1}$, and $I$ is an $m \times m$ identity matrix.
(3) *Graph weight setting*: Suppose $G = (V, E)$ is the graph constructed, where the vertex set $V$ consists of the data points $a_1, a_2, \ldots, a_n$ and $E$ is the edge set. $W$ is the corresponding similarity matrix. For the $L_1$-graph, we set $W_{ij} = x^i_j$ if $i > j$ and $W_{ij} = x^i_{j-1}$ if $i < j$. If the similarity measurement is considered for graph construction, we can set $W_{ij} = |x^i_j|$ if $i > j$ and $W_{ij} = |x^i_{j-1}|$ if $i < j$.

In fact, the reconstruction error sparsity has also been induced by the second step "sparse representation". We can transform the original optimization problem into

$$\min_{x^i_1, x^i_2} \|x^i_1\|_1 + \|x^i_2\|_1, \quad \text{s.t. } [B^i, I] \begin{bmatrix} x^i_1 \\ x^i_2 \end{bmatrix} = a_i \tag{3}$$

where $x^i_1 \in \mathbb{R}^{n-1}, x^i_2 \in \mathbb{R}^m$. From the above equation constraint, we derive that $x^i_2 = a_i - B^i x^i_1$. Hence, the original optimization problem is actually equivalent to

$$\min_{x^i_1} \|B^i x^i_1 - a_i\|_1 + \|x^i_1\|_1 \tag{4}$$

Recall that, in the Lasso method [11], we aim to solve the following optimization problem:

$$\min_x \|Ax - b\|_2^2 + \lambda \|x\|_1 \tag{5}$$

where $A \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$. Let $\lambda = 1$, and we can find that the sparse representation step of the $L_1$-graph construction is very similar to the Lasso method. Although the reconstruction error is measured with different norms, it is reasonable to believe that they have similar properties.

Besides, the $L_1$-graph is not applicable to large scale high-dimensional datasets because of its high computational costs. As we can see from the sparse representation step of $L_1$-graph construction, when faced with a dataset of size $n$ whose components are data vectors in $\mathbb{R}^m$, we have to deal with a matrix $A^i$ of size $m \times (n + m - 1)$ when solving the optimization problem given in Eq. (2). When $n$ and $m$ become relatively large (e.g. the PIE face dataset we used in this paper, which contains 11,554 $32 \times 32$ grayscale images and gives us a dataset of size 11,554 whose components are data vectors in $\mathbb{R}^{1,024}$), the time needed for constructing the entire $L_1$-graph is unbearable.

In Section 5, we will analyze the computational complexity of the original $L_1$-graph construction method, and compare it with the other graph construction methods. But to get an intuitive idea of its heavy computational burden, we first conduct an experiment on the PIE face dataset, in which we use the original $L_1$-graph construction method. To make this problem computationally tractable, we first reduce the dimension of the original dataset with principal component analysis, in which only those features that contribute to 0.5% or greater of the total variance is kept. The resulting dataset consists of 11,554 13-dimensional data point vectors. To construct the $L_1$-graph with the original $L_1$-graph construction method, we need to run the sparse representation step 11,554 times. Since our goal is to demonstrate the high computational costs of the original $L_1$-graph construction method, we only run the first 500 sparse representation steps and record the time needed for each of these steps. The specifications of the computer we used are: Intel Xeon CPU E5450 @3.00 GHz, 16 GB RAM. Experimental results show that the average time needed for one sparse representation step is 118 s, and the standard deviation for these 500 samples is 6.5. This means that we need approximately 2 min for each sparse representation step. If we want to complete the entire graph construction process, we will need about 23,108 min, which is more than 2 weeks. And this is only the time needed after dimension reduction. It is quite clear that the computational costs for the original $L_1$-graph construction method are unbearable.

## 3. Overcoming the shortcomings of $L_1$-graph

### 3.1. New regularization term

Following the discussion in Section 2, we can see that, in nature, the sparse representation step of the $L_1$-graph construction process is solving the optimization problem given by Eq. (4). As we have discussed in Section 1, when the data point $a_i$ is linearly reconstructed with other data points, it is reasonable to assume that the reconstruction coefficients of every data point and its nearest neighbors should be similar in value. Since in most cases, we do not have prior knowledge about the structure of the data set, we can induce such structured sparsity in a pairwise manner. Yet the advantage of $k$-NN graph over the full graph shows to us that not all pairwise relations between data points should be used. By considering only the most important edges, the $k$-NN graph excludes possible disturbance from irrelevant edges. At the same time, by reducing the number of edges considered, the $k$-NN graph also simplifies the problem.

Hence, we construct our regularization term as follows. Note that the construction of this regularization term also employs the $k$-NN method, which we will give a more detailed discussion in Section 3.2. Before we start the construction process, we choose a global parameter $k$, just the same as in the traditional $k$-NN graph. Then in the sparse representation step, for each reconstructed data point $a_i$, we determine its $k$ nearest neighbors, say $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$. Consider the following regularization term for the sparse

representation step:

$$\|x^i\|_1 + \gamma \sum_{1 \le j < l \le k} w_{jl} |x^i_{i_j} - x^i_{i_l}| \tag{6}$$

where $w_{jl}$ is the weight for term $|x^i_{i_j} - x^i_{i_l}|$. We call $\|x^i\|_1$ the Lasso part, and $\sum_{1 \le j < l \le k} w_{jl} |x^i_{i_j} - x^i_{i_l}|$ the $k$-NN fused Lasso part (which we will denote as KFL part for the rest of the paper). $\gamma$ is a parameter introduced for the purpose of trade-off between these two parts, but is not actually used because of our reformulation strategy described in Section 3.3. This regularization term can harness the information provided by traditional measurement of similarity and exploit the local structure across the dataset to induce structured sparsity in the sparse representation step. Note that this new regularization term is indeed an extension of the fused Lasso method [13], which is one of the many extensions to the traditional Lasso method [12–14]. This is also the reason for the name of our new graph.

### 3.2. k-NN method and kernel method

In this section, we describe in detail why and how we employ the $k$-NN method and the kernel method in the graph construction process to handle large scale high-dimensional datasets. As we have mentioned in Section 3.1, the $k$-NN graph simplifies the problem and achieves great advantage over the full graph by excluding irrelevant information with the $k$-NN method. The $L_1$-graph seeks to determine the neighbors of each data point automatically by the sparse representation step, yet as the following experiments shows (see results in Table 1), in most cases, a large proportion of the neighbors selected this way are also among the $k$-nearest neighbors of the reconstructed data point, only with $k$ being relatively large.

The experiment is conducted as follows. We still use the PIE face dataset, and first reduce its dimensions in the way described at the end of Section 2. We then construct the $L_1$-graph on this dataset with the original $L_1$-graph construction method. As pointed out in Section 2, the computational costs of this process are extremely high. Since our goal is to show that it is reasonable to employ the $k$-NN method in the graph construction process, we still run only the first 500 sparse representation steps, and get 500 vectors, which represents the neighbors selected by the $L_1$-graph and the corresponding edge weights in the graph. Given the value of parameter $k$, we want to know how many neighbors selected by the original $L_1$-graph are also among the $k$-nearest neighbors of the reconstructed data point, and more importantly, how many neighbors with large edge weights, which play the most important role in the subsequent machine learning tasks, are among the $k$-nearest neighbors. To know this, we define the $k$-NN ratio without threshold, which is the number of the selected (by the original $L_1$-graph) neighbors that are among the $k$-nearest neighbors of reconstructed data point divided by the number of all the selected (by the original $L_1$-graph) neighbors. We also define the $k$-NN ratio with threshold $t$, which is the number of the selected (by the original $L_1$-graph) neighbors that are among the $k$-nearest neighbors of reconstructed data point and have edge weights larger than $t$ divided by the number of all the selected (by the original $L_1$-graph) neighbors whose edge weights are larger than $t$ in order to see how many of the most important selected neighbors fall in the $k$-nearest neighbors. The $k$-NN ratios for different $k$ and different thresholds are reported in Table 1.

**Table 1**
Average $k$-NN ratio (the average of the $k$-NN ratios computed from the first 500 sparse representation steps) on the PIE face dataset with different value of parameter $k$ and different threshold setting. The variable $t$ denotes the threshold.

| | No threshold (%) | $t$=0.01 (%) | $t$=0.001 (%) |
|---|---|---|---|
| $k$=50 | 20.10 | 39.58 | 36.55 |
| $k$=100 | 27.17 | 46.75 | 43.87 |
| $k$=200 | 35.73 | 54.58 | 52.35 |
| $k$=500 | 50.36 | 66.01 | 63.95 |

We can see from Table 1 that when $k$=50, the average $k$-NN ratio is already quite high, especially for $t$=0.01. When $k$ becomes larger, the average $k$-NN ratio also increases greatly. When $k$=500, more than half of the neighbors selected by the $L_1$-graph is among the $k$-nearest neighbors, and nearly 70% of the most important neighbors selected (those with edge weights larger than 0.01) is among the $k$-nearest neighbors. This means that quite a lot of the important neighbors selected by the $L_1$-graph are also among the $k$-nearest neighbors of the reconstructed data point. Note that as $k$ increases, the corresponding $k$-NN ratio will also increase. In fact, when $k = n-1$ (which means we are using all the data points), the $k$-NN ratio will be exactly 1. As a result, we cannot predict the performance of the graph construction method based on its $k$-NN ratio values for different $k$ when the non-$k$-NN data points are dropped, since we can always increase $k$ to get a higher $k$-NN ratio, but increasing the value of $k$ will not necessarily result in a better performance, as shown, for example, in our parameter sensitivity analysis in Section 6.5. We also observe that although the $k$-NN ratio increases as $k$ increases, it is less than 1 when $k$ is not so large. This shows that the sparse representation method excludes some neighbors (possibly irrelevant data points) selected by the $k$-NN method. This demonstrates the difference between the two ways of measuring similarity (i.e. sparse representation and the traditional ones, as we have mentioned at the beginning of the introduction), and suggests that we may also exclude the irrelevant points selected by sparse representation by restricting our attention on the $k$-nearest neighbors in the sparse representation step, which means combining the $k$-NN method with sparse representation may potentially improve the performance of the $L_1$-graph. This observation, along with the relatively high $k$-NN ratio, indicates to us that it is completely reasonable to employ the $k$-NN method to handle large scale datasets.

As a result, it will be much more computationally efficient if we regress each data point only on its $k$ nearest neighbors and set the reconstruction coefficient of all the other data points that are not in the $k$-nearest neighbors to be zero. Similarly, we can use the $k$-NN method in the Lasso part of our new regularization term. Since we have already used the $k$-NN method in the KFL part, we need to introduce two parameters $k_1$ (for the Lasso part) and $k_2$ (for the KFL part). We also set the reconstruction coefficient of all the data points that are not in the $k_1$-nearest neighbors to be zero, so the data points involved in the KFL part should also be in the Lasso part, which makes $k_2 \le k_1$.

The kernel method is used to deal with the high-dimensionality of the datasets. Recall that, in the $L_1$-graph, the authors originally considered the optimization problem

$$\min_{x^i} \|x^i\|_1 \quad \text{s.t. } B^i x^i = a_i \tag{7}$$

where $B^i = [a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n]$. Multiply each side of the equation constraint by the transpose of $B^i$, and the equation constraint becomes

$$[\langle a_j, a_k \rangle]_{(n-1)\times(n-1)} x^i = [\langle a_j, a_i \rangle]_{(n-1)\times 1} \tag{8}$$

where $j, k = 1, \ldots, i-1, i+1, \ldots, n$, $\langle a_j, a_k \rangle = a_j^T a_k$. $\langle a_j, a_k \rangle$ can be seen as the inner product of $a_j, a_k$. So if we have some kind of kernel matrix derived from the datasets, we can use it instead of the original data point vectors to construct the matrix $B^i$. If we combine the kernel method with the $k$-NN method described above, we can get a $B^i$ of size $k \times k$, and thus a $A^i = [B^i, I]$ of size $k \times 2k$, which is completely independent of the dimensionality of

the data point vectors. The $L_1$-graph construction process with $k$-NN method and kernel method is summarized as follows:

(1) **Initialization**: Choose the value of parameter $k$. The input now is a kernel matrix $K=[K_{ij}]_{n\times n}$ derived from the set of data points $a_1, a_2, \ldots, a_n \in \mathbb{R}^m$.

(2) **Sparse representation**: For each data point $a_i$, we determine its $k$-nearest neighbors $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$. Let

$$B^i=[K_{i_p i_q}]_{k\times k}, \quad p,q=1,2,\ldots,k \tag{9}$$

Define $A^i=[B^i, I]\in\mathbb{R}^{k\times 2k}$, where $I$ is an $k\times k$ identity matrix. Define $K_i=[K_{i_1 i}, K_{i_2 i}, \ldots, K_{i_k i}]^T$, and we solve the optimization problem for a $2k$-dimensional vector $x^i$

$$\min_{x^i}\|x^i\|_1, \quad \text{s.t. } A^i x^i = K_i \tag{10}$$

(3) *Graph weight setting*: Suppose $G=(V,E)$ is the graph constructed, where the vertex set $V$ consists of the data points $a_1, a_2, \ldots, a_n$ and $E$ is the edge set. $W$ is the corresponding similarity matrix. For the $L_1$-graph, we set $W_{ii_j}=x_j^i$, $j=1,2,\ldots,k$. If the similarity measurement is considered for graph construction, we can set $W_{ii_j}=|x_j^i|$. The rest of $W$ is set to be zero.

### 3.3. Regularization term reformulation

In this section, we reformulate the new regularization term into matrix form to fit it into the graph construction process. Suppose $k_2\le k_1$, and $a_{i_1}, \ldots, a_{i_{k_2}}, \ldots, a_{i_{k_1}}$ is the $k_1$-nearest neighbors of $a_i$. Since the reconstruction coefficients of all the data points that are not in the $k_1$-nearest neighbors of $a_i$ are set to be zero, we can measure the reconstruction error with

$$\|[a_{i_1}, a_{i_2}, \ldots, a_{i_{k_1}}]x^i - a_i\|_1 \tag{11}$$

Then the corresponding regularization term will be

$$\|x^i\|_1 + \gamma \sum_{1\le j<l\le k_2} w_{jl}|x_j^i - x_l^i| \tag{12}$$

where $x^i$ is a $k_1$-dimensional vector. Consider the KFL part $(\sum_{1\le j<l\le k_2} w_{jl}|x_j^i - x_l^i|)$ of the regularization term. We construct a sparse matrix

$$C^i=(c_{ij})\in\mathbb{R}^{k_2(k_2-1)/2\times k_1} \tag{13}$$

with

$$c_{(2k_2-j)(j-1)/2+(l-j),j} = w_{jl} \tag{14}$$

$$c_{(2k_2-j)(j-1)/2+(l-j),l} = -w_{jl} \tag{15}$$

and the rest of its entries all being zeros. Then we have

$$\sum_{1\le j<l\le k_2} w_{jl}|x_j^i - x_l^i| = \|C^i x^i\|_1 \tag{16}$$

In our paper, for data point $a_i$, we let the weight $w_{jl}=K_{i_j i_l}$.

In the $L_1$-graph, there may exist noise in the reconstruction process (i.e. the equation constraint $B^i x^i = a_i$ in the optimization problem (7) may not strictly hold). The authors added an identity matrix to $B^i$ to represent the noise term so that the error term $\|B^i x^i - a_i\|_1$ can also be considered in the graph construction process.

Inspired by this strategy, for a given data point $a_i$, we first follow the method proposed in Eqs. (13)–(16) and reformulate the KFL part of the regularization term as $\sum_{1\le j<l\le k_2} w_{jl}|x_j^i - x_l^i| = \|C^i x^i\|_1$. Furthermore, we define the matrix

$$M^i=\begin{bmatrix} B^i & I & 0 \\ C^i & 0 & I \end{bmatrix} \tag{17}$$

Here, matrix $B^i$ is given by Eq. (9)(with $k=k_1$) with the help of kernel method. Hence, with parameters $k_1$ and $k_2$ fixed, the matrix $M^i$ is of size $(k_1+k_2(k_2-1)/2)\times(2k_1+k_2(k_2-1)/2)$. Finally, let $K_i=[K_{i_1 i}, K_{i_2 i}, \ldots, K_{i_{k_1} i}]^T$, and we get a generalized version of the sparse representation step by solving the optimization problem

$$\min_{x^i,\epsilon^i,\eta^i}\|x^i\|_1 + \|\epsilon^i\|_1 + \|\eta^i\|_1 \quad \text{s.t. } M^i\begin{bmatrix} x^i \\ \epsilon^i \\ \eta^i \end{bmatrix} = \begin{bmatrix} K_i \\ 0 \end{bmatrix} \tag{18}$$

where $x^i\in\mathbb{R}^{k_1}$, $\epsilon^i\in\mathbb{R}^{k_1}$, and $\eta^i\in\mathbb{R}^{k_2(k_2-1)/2}$. This problem is solved by the classical primal-dual interior point method [15]. The weight matrix $W$ of the obtained $k$-NN fused Lasso graph can be set in the same way as the original $L_1$-graph.

## 4. *k*-NN fused Lasso graph construction

Finally, we summarize the graph construction process for our $k$-NN fused Lasso graph as follows:

(1) *Initialization*: Choose the value of parameter $k_1, k_2$, s.t. $k_2\le k_1$. The input now is a kernel matrix $K=[K_{ij}]_{n\times n}$ derived from the set of data points $a_1, a_2, \ldots, a_n\in\mathbb{R}^m$.

(2) *Sparse representation*: For each data point $a_i$, we determine its $k_1$ nearest neighbors $a_{i_1}, \ldots, a_{i_{k_2}}, \ldots, a_{i_{k_1}}$. We construct the new regularization term with Eq. (12) and get the matrix $C^i$ with Eqs. (13)–(16). Let $B^i=[K_{i_p i_q}]_{k_1\times k_1}$, where $p,q=1,2,\ldots,k_1$. Define the following matrix:

$$M^i=\begin{bmatrix} B^i & I & 0 \\ C^i & 0 & I \end{bmatrix} \tag{19}$$

where $I$ is identity matrix. Define $K_i=[K_{i_1 i}, K_{i_2 i}, \ldots, K_{i_{k_1} i}]^T$, and we solve the optimization problem

$$\min_{x^i,\epsilon^i,\eta^i}\|x^i\|_1 + \|\epsilon^i\|_1 + \|\eta^i\|_1 \quad \text{s.t. } M^i\begin{bmatrix} x^i \\ \epsilon^i \\ \eta^i \end{bmatrix} = \begin{bmatrix} K_i \\ 0 \end{bmatrix} \tag{20}$$

where $x^i\in\mathbb{R}^{k_1}$, $\epsilon^i\in\mathbb{R}^{k_1}$, and $\eta^i\in\mathbb{R}^{k_2(k_2-1)/2}$.

(3) *Graph weight setting*: Suppose $G=(V,E)$ is the graph constructed, where the vertex set $V$ contains $a_1, a_2, \ldots, a_n$ and $E$ is the edge set. $W$ is the corresponding similarity matrix. For the $k$-NN fused Lasso graph, we set $W_{ii_j}=x_j^i, j=1,2,\ldots,k_1$. If the similarity measurement is considered for graph construction, we can set $W_{ii_j}=\left|x_j^i\right|$. The rest of $W$ is set to be zero.

It should be noted that the greatest advantage of our $k$-NN fused Lasso graph over the traditional $k$-NN graph is its better performance, which will be presented in the experimental results section. Moreover, as shown in the next section on computational complexity analysis, for dataset of moderate size, our $k$-NN fused Lasso graph may be more computationally intensive, but when dealing with really large dataset, it is possible that our $k$-NN fused Lasso graph will lead to less computational cost than the traditional $k$-NN graph.

## 5. Computational complexity analysis

In this section, we give a theoretical analysis of the computational complexity of the graph construction methods involved in this paper. Four different graph construction methods are used in

this paper, namely the $k$-NN graph, the original $L_1$-graph (which is described in Section 2), the $L_1$-graph with $k$-NN method and kernel method (which is described in Section 3.2 and used in the experiments in Section 6 as $L_1$-graph. We will refer to this as the new $L_1$-graph for convenience), and our new $k$-NN fused Lasso graph (which is described in Section 4). We assume that the size of the dataset we are interested in is $n$, and the dimension of each data point in this dataset is $m$. No parameter is needed for the original $L_1$-graph. For the new $L_1$-graph, we assume the value of the parameter is $k$. For the $k$-NN fused Lasso graph, we assume the values of the two parameters are $k_1$ and $k_2$.

To discuss and compare their computational complexity, first we need to look at an optimization problem that is central to the last three methods. In the sparse representation step of these methods, we try to solve an optimization problem of the form

$$\min_x \|x\|_1, \quad \text{s.t. } Ax = a \tag{21}$$

where $x \in \mathbb{R}^p, a \in \mathbb{R}^q$, and $A$ is a $q \times p$ matrix.

This is a classical $L_1$-minimization problem, and lots of methods have been proposed to solve it (see [19] for a review). In this paper, we employed the basic primal-dual interior point method, whose complexity is bounded by $O(\sqrt{p}p^3) = O(p^{\frac{7}{2}})$ as pointed out in [19]. For more general case, we can assume that the computational complexity of the solver we employed is $O(h(p,q))$, where $h(p,q)$ is a function depending on the method we use to solve the $L_1$-minimization problem (21).

In the last three methods, the computational complexity is dominated by the sparse representation step. As a result, with the above assumption, it is easy to see that the computational complexities of the original $L_1$-graph, the new $L_1$-graph, and the $k$-NN fused Lasso graph are approximately $O(nh(n+m-1,m))$, $O(nh(k,2k))$ and $O(nh(2k_1 + k_2(k_2-1)/2, k_1 + k_2(k_2-1)/2))$ respectively. For the $k$-NN graph, since we take a kernel matrix as input, we only need to pick the $k$ largest elements of the $n$ rows of the matrix. This can be done by sorting each row of the matrix, and has a worst case complexity of $O(n \cdot n^2) = O(n^3)$. If we use, for example, the quicksort algorithm, we can get an average complexity of $O(n \cdot n \log(n)) = O(n^2 \log(n))$.

Most of the time, the $L_1$-minimization problem (21) has to be solved iteratively. In each iteration, the worst case complexity may be up to $O(n^3)$, and it may take quite a few iterations before the algorithm can finally converge (e.g. the primal-dual interior point method employed in our experiments takes $O(\sqrt{n})$ iterations to converge [19]). Although many new algorithms [19] have been proposed, in terms of computational complexity on large complex datasets, the function $h(p,q)$ is usually much worse than the $n^2$ we have in the $k$-NN graph case. This is the reason for which the computational costs of the original $L_1$-graph are unbearable when it comes to large complex datasets, as shown in the experiments in Section 2, and this is also the reason for which we employ the $k$-NN method and kernel method. Note that in the case of the new $L_1$-graph and the $k$-NN fused Lasso graph, as a result of the $k$-NN method and kernel method, the arguments of the function $h(p,q)$ depend only on the values of the parameters in the model, which are significantly smaller than the size of the dataset $n$ and have nothing to do with the potentially high dimension of each data point. Consequently, our new graph construction method is capable of dealing with very large and complex datasets. If we choose $k_1 = k$, the new $k$-NN fused Lasso graph is slightly more computationally intensive because of the incorporated structured sparsity information (which is reflected by the parameter $k_2$), but experiments show that a small $k_2$ (usually $\leq 20$) suffices to get better performance than the new $L_1$-graph. Also, the structured sparsity information enables us to learn more with fewer neighbors, i.e. the optimal $k_1$ in $k$-NN fused Lasso graph is often smaller than the optimal $k$ in the new $L_1$-graph, which makes the better performing $k$-NN fused Lasso graph more computationally efficient than the new $L_1$-graph in some cases, and certainly more efficient than the original $L_1$-graph. If $n$ becomes really large, even constructing the $k$-NN graph is time-consuming, but if we choose $k_1$ and $k_2$ to be reasonably small, constructing the $k$-NN fused Lasso graph may still be feasible and efficient.

## 6. Experimental results

In this section, we conduct a variety of experiments to demonstrate the effectiveness of our $k$-NN fused Lasso graph and also its advantage over the traditional $L_1$-graph. In this paper, we focus on testing our $k$-NN fused Lasso graph in the task of spectral clustering, regardless of many other graph-based machine learning tasks.

### 6.1. Datasets

In our experiments, four image datasets are used for performance evaluation. The Scene dataset contains eight scene categories from MIT [16], including four man-made scenes and four natural scenes. The total number of images is 2688. The size of each image in this Scene dataset is $256 \times 256$ pixels. The Corel dataset consists of 2000 images taken from 20 CD-ROMs published by COREL Corporation. Each COREL CD-ROM contains 100 images representing a distinct concept. Therefore, the dataset has 20 thematically diverse image categories, each containing 100 images. All the images are of size $384 \times 256$ or $256 \times 384$. The Caltech 256 dataset [21] is a challenging set of 257 object categories containing a total of 30,607 images. In our experiment, we discard the clutter category and use only the other 256 categories, which results in a dataset of 29,780 images. The PIE face dataset is a large scale high-dimensional dataset. It consists of 41,368 images of 68 people, where each person is under 13 different poses, 43 different illumination conditions, and with four different expressions. In our experiment, we choose a subset of this dataset, which only contains five near frontal poses (C05, C07, C09, C27, C29) and all the images under different illuminations and expressions. So, there are about 170 images of size $32 \times 32$ for each individual, which gives us a face dataset of size 11,554.

### 6.2. Experimental setting

In this paper, we focus on comparing different graph construction methods in spectral clustering problems. Spectral clustering is one of the most popular modern clustering algorithms, and there are several variations of this algorithm. In this paper, we employ the widely used normalized spectral clustering algorithm [1], which is outlined as follows:

(1) Inputs: Graph similarity matrix $W \in \mathbb{R}^{n \times n}$ and number $c$ of clusters to construct. Symmetrize the matrix by setting $W = (W + W^T)/2$.
(2) Compute the graph Laplacian matrix $L = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$, where $D = [d_{ij}]$ is a diagonal matrix with the diagonal element $d_{ii} = \sum_{j=1}^{n} w_{ij}$.
(3) Compute $c_1, c_2, \ldots, c_c$, the eigenvectors of $L$ corresponding to the $c$ largest eigenvalues, and get a matrix $C = [c_1, c_2, \ldots, c_c]$.
(4) Regard each row of matrix $C$ as a point in $\mathbb{R}^c$, and cluster them into $c$ clusters via the k-means method.
(5) Assign data point $a_i$ to the cluster $j$ if the $i$th row of the matrix $C$ is assigned to the cluster $j$.

In this paper, we are dealing with several large complex image datasets. As we have pointed out in Section 2, the original $L_1$-graph

construction method is not applicable to such datasets because of its unbearable computational costs. As a result, we use the modified $L_1$-graph construction method described in Section 3.2, in which the $k$-NN method and kernel method are used to handle large complex datasets.

Since we use kernel method in our graph construction process, we need to derive a kernel matrix from the dataset to serve as input to our algorithm. We adopt two different approaches to kernel matrix computation. For the three natural image datasets (i.e. Scene, Corel and Caltech 256), we compute the spatial Markov kernel matrix [17] based on 400 visual words (i.e. 400 features), just the same as [2]. For the PIE face dataset, we use the Gaussian Kernel and set $K_{ij} = \exp(-\|a_i - a_j\|_2^2/\beta)$, where $\beta$ is a parameter and is set to be the same for different graphs.

The normalized mutual information (NMI) [18] is used to evaluate the performance of spectral clustering on different graphs. Suppose $X$ is the clustering result and $Y$ is the known sample label vector. Let $p(x)$ and $p(y)$ denote the marginal probability mass functions of $X$ and $Y$, and let $p(x,y)$ be the joint probability mass function of $X$ and $Y$. Suppose $H(X), H(Y)$ and $H(X,Y)$ denote the entropies of $p(x), p(y)$ and $p(x,y)$ respectively. Then the normalized mutual information NMI is given by

$$NMI(X,Y) = \frac{H(X) + H(Y) - H(X,Y)}{\max(H(X), H(Y))} \tag{22}$$

It is obvious that the normalized mutual information NMI takes values in $[0, 1]$. The higher its value is, the better a graph performs.

Our experiments are conducted as follows. For each dataset, we first derive the kernel matrix $K$ from the data points $a_1, a_2, \ldots, a_n$, and construct three different graphs for the normalized spectral clustering algorithm [1]: $k$-NN graph, $L_1$-graph, and $k$-NN fused Lasso graph. The $k$-NN graph is constructed with the derived kernel matrix $K$, and the graph construction process for the $L_1$-graph and the $k$-NN fused Lasso graph is given in Sections 3.2 and 4, respectively. We follow the spectral clustering algorithm outlined at the beginning of this section, and compute the corresponding normalized mutual information NMI by Eq. (22) with the clustering result and the known sample label vector. The parameter $k$ in $k$-NN graph construction is carefully tuned for the best performance, and the parameter $k$ in $L_1$-graph construction and the parameters $k_1, k_2$ in $k$-NN fused Lasso graph construction are only roughly tuned.

### 6.3. Results on the image datasets

We conduct the experiments on the four image datasets, and the results are listed in Table 2. The immediate observation is that our $k$-NN fused Lasso graph can always achieve the best performance. This may be due to the fact that our method has successfully incorporated the structured sparsity into $L_1$-graph construction. We give more detailed explanation as follows. On the one hand, although the regularization term used to induce the structured sparsity is defined based on the $k$-NN method, our $k$-NN fused Lasso graph is shown to outperform the $k$-NN graph, since the main part of our graph construction process is still the sparse representation step. Here, it should be noted that the effectiveness of sparse representation for $L_1$-graph construction has been verified by the significant gain over $k$-NN graph achieved by the $L_1$-graph. In fact, the improved performance of our $k$-NN fused Lasso graph over the $k$-NN graph provides further verification of the effectiveness of sparse representation for $L_1$-graph construction. On the other hand, our $k$-NN fused Lasso graph is shown to indeed benefit from the structured sparsity induced by the new regularization term, given that it achieves 2.3% average gain (not relative) over the $L_1$-graph.

Note that for such datasets, especially the PIE face dataset with 11,554 instances and the Caltech 256 dataset with 29,780 instances, the computational costs for the original $L_1$-graph are unbearable. However, with $k$-NN method and kernel method, the $L_1$-graph can be applied to even larger datasets, and according to our experimental results reported in Table 2, it can still achieve great advantage over the $k$-NN graph.

**Table 2**
Clustering accuracies (Normalized Mutual Information/NMI) for spectral clustering based on $k$-NN graph, $L_1$-graph and $k$-NN fused Lasso graph on four image datasets as well as the Doubled Scene dataset. Average* is the average NMI computed without the results on the Doubled Scene dataset.

| Datasets | Graph-based spectral clustering | | |
|---|---|---|---|
| | $k$-NN graph | $L_1$-graph | $k$-NN fused Lasso graph |
| Scene | 0.612 | 0.619 | 0.632 |
| Doubled scene | 0.612 | 0.005 | 0.638 |
| Corel | 0.555 | 0.570 | 0.608 |
| Caltech 256 | 0.288 | 0.306 | 0.321 |
| PIE face | 0.376 | 0.383 | 0.412 |
| Average | 0.489 | 0.377 | 0.522 |
| Average* | 0.458 | 0.470 | 0.493 |

### 6.4. Further comparison between $L_1$-graph and $k$-NN fused Lasso graph

As we have mentioned in the introduction, the $L_1$-graph has several potential drawbacks. According to the discussion in [10], in extreme cases where each sample is simply duplicated to generate a new dataset of double size, the $L_1$-graph tends to only connect these duplicated pairs and thus may fail to capture the valuable information from the original set of data points. We have given a direct illustration of this phenomenon in Fig. 1 in the Introduction. In this section, we conduct a group of experiments on the Scene dataset to further illustrate this drawback of the $L_1$-graph and demonstrate the advantage of $k$-NN fused Lasso graph over the $L_1$-graph in this aspect.

We first make exact duplicate of each data point in the original Scene dataset and generate a new dataset (called doubled Scene) of double size. We then construct both $L_1$-graph and $k$-NN fused Lasso graph on this doubled dataset. The similarity matrices for the two graph construction methods on the doubled Scene dataset are illustrated in Fig. 2. Comparing this figure with Fig. 1, we can find that, on the complex Scene dataset, the $L_1$-graph performs exactly the same as it does on the small soybean dataset, and connects only the duplicated data point pairs (with all these edge weights being 1 approximatively). When it comes to the $k$-NN fused Lasso graph, we can still see that not only the duplicated data point pairs but also the unduplicated pairs are connected. Although the block structure is not as obvious as that on the small and easy soybean dataset, it is clear that the structured sparsity is induced, and from the experimental results reported in Table 2, we can see the effectiveness of our new graph construction method.

We further make attempt to demonstrate that the edges between unduplicated pairs in the $k$-NN fused Lasso graph constructed by our new graph construction method do help to improve the performance of spectral clustering. We again perform spectral clustering on the doubled Scene dataset. From the comparison results listed in Table 2, we can see that the Normalized Mutual Information NMI for $L_1$-graph on the doubled Scene dataset is significantly smaller than that on the original Scene dataset. In fact, it is quite close to zero, which clearly demonstrates the drawbacks of $L_1$-graph. In contrast, the performance of $k$-NN graph and $k$-NN fused Lasso graph remains unchanged or even slightly increased when the dataset is doubled. Here, it should be noted that the NMI for our $k$-NN fused Lasso graph is still 2.6% (not relative) higher than that of the $k$-NN graph.

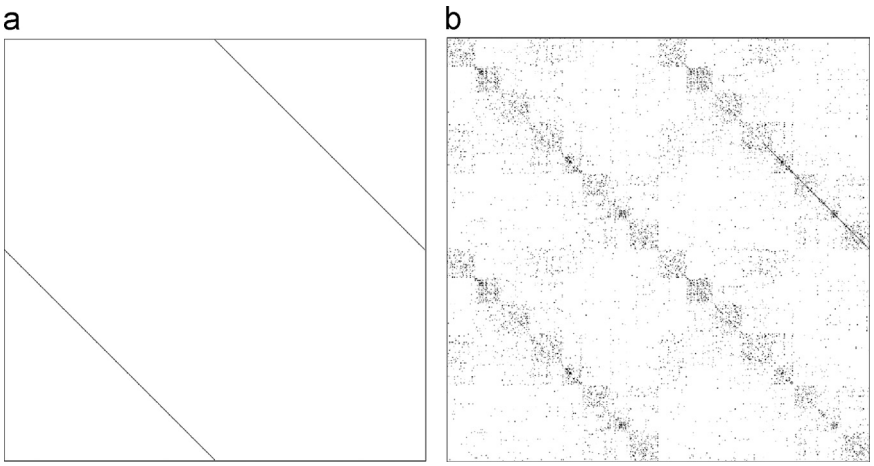


**Fig. 2.** Comparison between the similarity (i.e. weight) matrices of the $L_1$-graph and our $k$-NN fused Lasso graph on the doubled Scene dataset. For illustration purpose, the first and second half of the new dataset are identical copies of the original dataset, and each copy of the original dataset is rearranged such that data points within a class appear consecutively. More notably, the darker is a pixel, the larger is the similarity. (a) $L_1$-graph and (b) our graph.

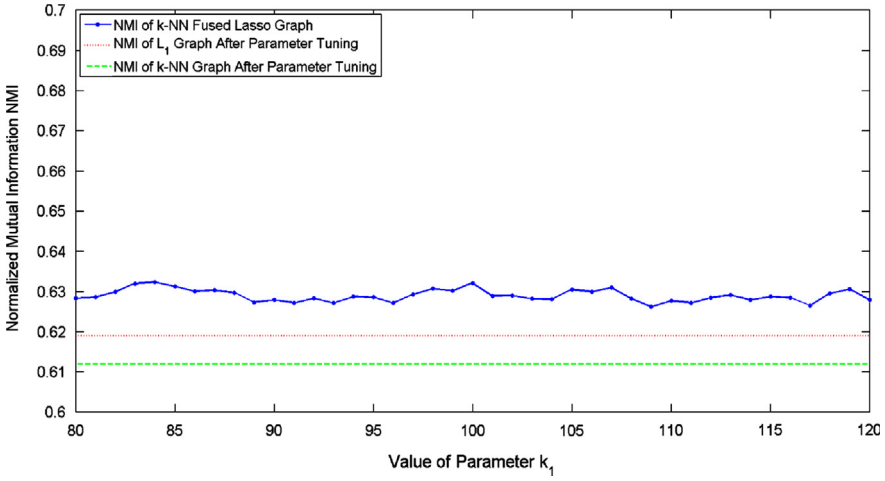


**Fig. 3.** Parameter sensitivity test for our $k$-NN fused Lasso graph with respect to $k_1$ on the Scene dataset (with fixed $k_2 = 5$).

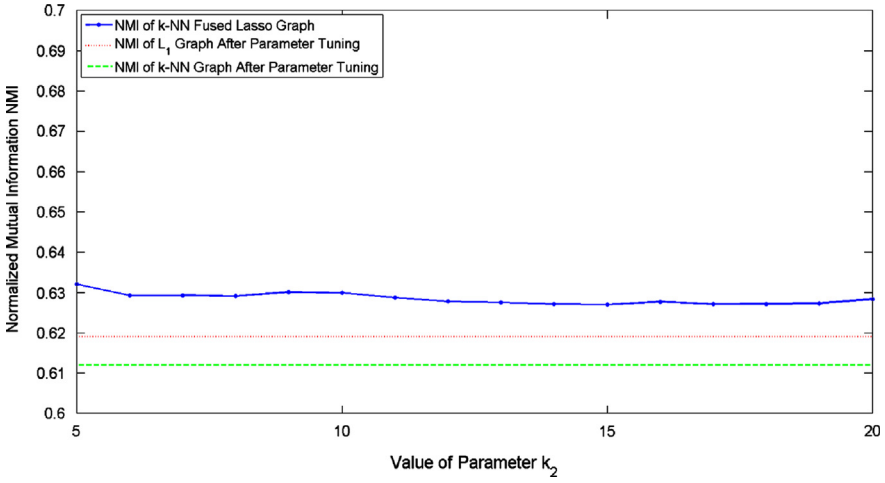


**Fig. 4.** Parameter sensitivity test for our $k$-NN fused Lasso graph with respect to $k_2$ on the Scene dataset (with fixed $k_1 = 100$).

### 6.5. *Parameter sensitivity test for k-NN fused lasso graph*

In the above experiments, we have successfully demonstrated the effectiveness of our new $k$-NN fused Lasso graph. However, our graph construction method has two $k$-NN parameters ($k_1$ and $k_2$) to be determined. In this section, we conduct a group of experiments on the four image datasets to test the parameter sensitivity of our graph construction method. The experimental results are reported in Figs. 3–10. Note that with rough tuning of parameters, we achieve the highest NMI value with $k_1 = 100$ and $k_2 = 5$ on

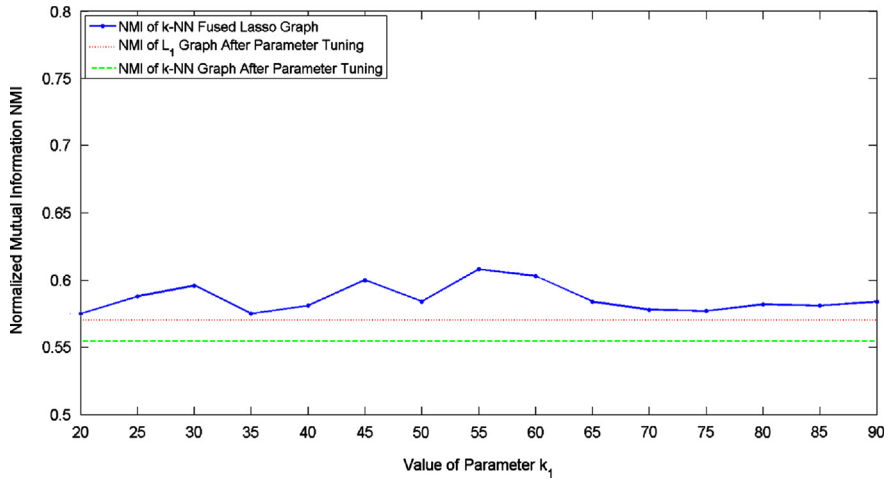


**Fig. 5.** Parameter sensitivity test for our $k$-NN fused Lasso graph with respect to $k_1$ on the Corel dataset (with fixed $k_2 = 20$).

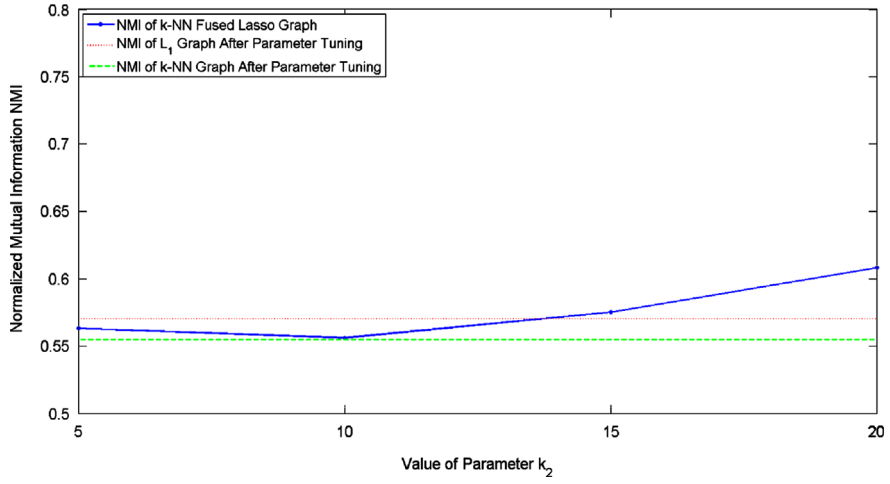


**Fig. 6.** Parameter sensitivity test for our $k$-NN fused Lasso graph with respect to $k_2$ on the Corel dataset (with fixed $k_1 = 55$).

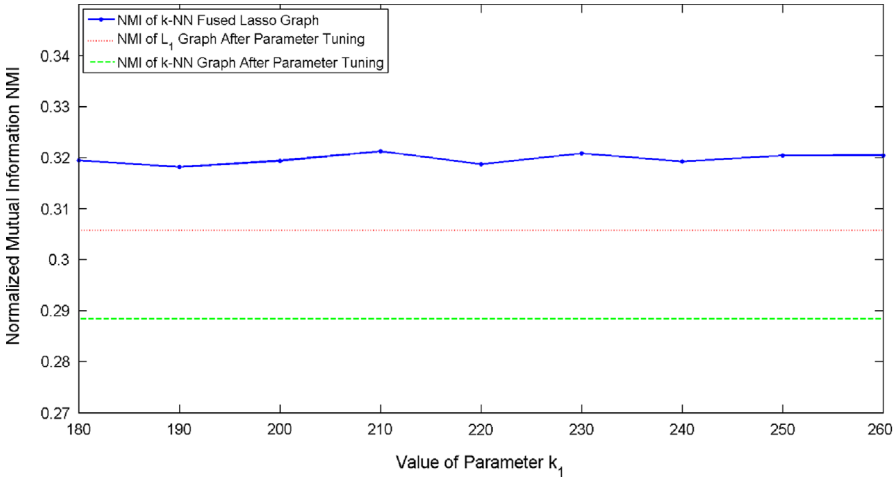


**Fig. 7.** Parameter sensitivity test for our $k$-NN fused Lasso graph with respect to $k_1$ on the Caltech 256 dataset (with fixed $k_2 = 20$).

Scene, with $k_1 = 55$ and $k_2 = 20$ on Corel, with $k_1 = 210$ and $k_2 = 20$ on Caltech 256, and with $k_1 = 10$ and $k_2 = 3$ on the PIE face dataset. So in this experiment, we fix $k_1$ ($k_2$) at its optimal value and let the value of the other parameter vary. We can find that on both Scene and Caltech 256, our graph construction method is quite robust with respect to the two $k$-NN parameters $k_1$ and $k_2$, and the performance of our graph is consistently better than that of the $k$-NN graph and the $L_1$-graph, no matter which values the two parameters take. Our method is not so robust on the other two datasets, but we can see that for most values of $k_1$, our method achieves higher performance than the other two existing methods. When it comes to $k_2$, the fluctuation is greater. However, it is worth noting that in our new

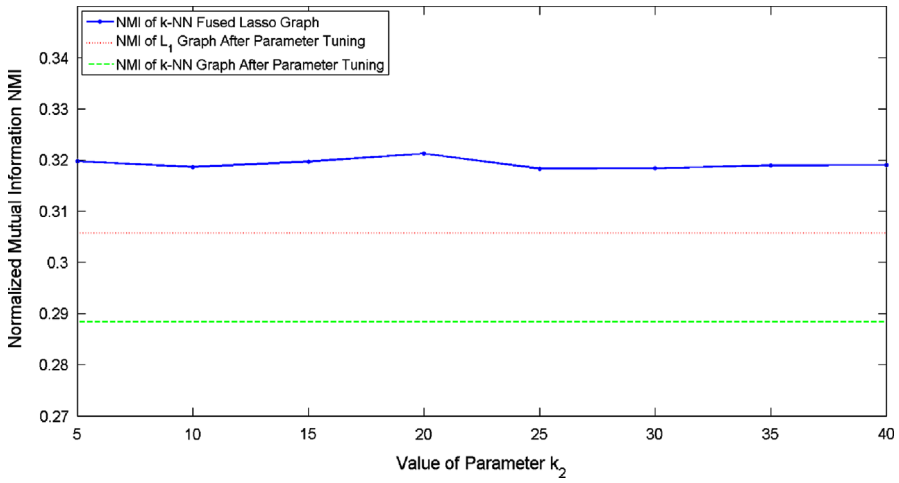


**Fig. 8.** Parameter sensitivity test for our $k$-NN fused Lasso graph with respect to $k_2$ on the Caltech 256 dataset (with fixed $k_1 = 210$).

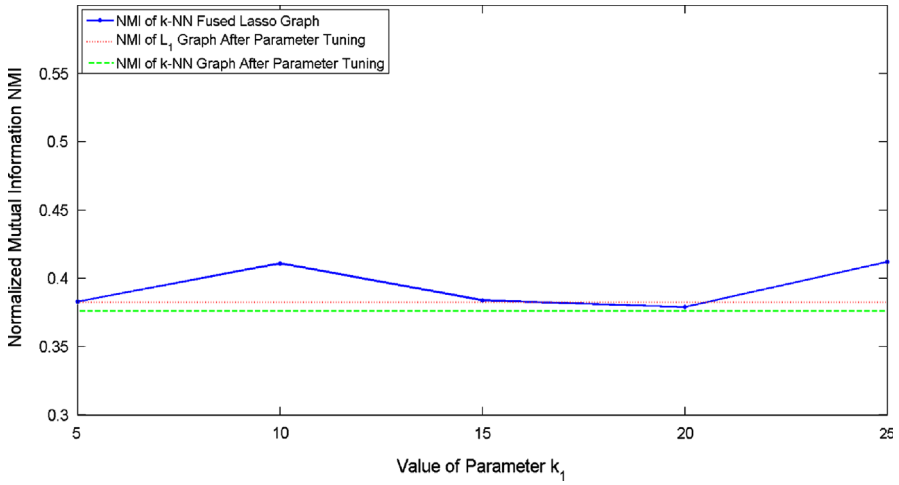


**Fig. 9.** Parameter sensitivity test for our $k$-NN fused Lasso graph with respect to $k_1$ on the PIE face dataset (with fixed $k_2 = 3$).

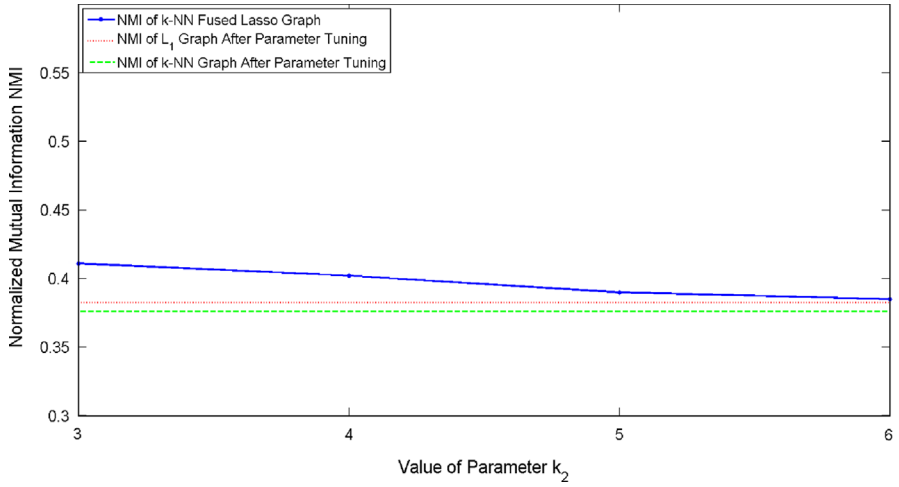


**Fig. 10.** Parameter sensitivity test for our $k$-NN fused Lasso graph with respect to $k_2$ on the PIE face dataset (with fixed $k_1 = 10$).

regularization term, the Lasso part has $k_1$ terms ($|x_j^i|$, $j = i_1, \ldots, i_{k_1}$) while the KFL part has $\Theta(k_2^2)$ ($k_2(k_2-1)/2$ to be more specific) terms ($w_{jl}|x_j^i - x_l^i|$, $1 \le j < l \le k_2$). On both Corel and PIE, the optimal value of $k_1$ is relatively small. As a result, when $k_2$ changes, the magnitude of the KFL part will change greatly compared with the Lasso part (because the KFL part has $\Theta(k_2^2)$ terms), which makes our method relatively sensitive to the value of $k_2$ on these two datasets. But we should see that in this case, where the optimal value of $k_1$ is relatively small, since $k_2 \le k_1$, the search space of $k_2$ is also quite small. Therefore, it still requires only a small number of trials before we can reach the optimal value of $k_2$, regardless of the fact that our method is not very robust with respect to $k_2$.

Now, based on the above observations and our experiences with the new graph construction method, we want to give some rules of thumb on how to choose the values of $k_1$ and $k_2$. The first step should be to choose the search range for $k_1$. Intuitively, $k_1$ indicates how large a neighborhood we want to consider, and should be closely related with dataset characteristics. For example, from our experimental results, we can see that the optimal value of $k_1$ on the fairly large PIE dataset is indeed the smallest. This demonstrates to us that local structure is very important in the PIE dataset, and a small $k_1$ will probably yield good results. To get an idea of how large a neighborhood we need, we can first take a look at the performance of the $k$-NN graph with different $k$, and the optimal value $k_{opt}$ of $k$ for the $k$-NN graph should be a good starting point for the search of the optimal value of $k_1$. However, we should note that although we will exclude some additional data points in the sparse representation step, $k_1$ is not necessarily larger than $k_{opt}$, since with the induced structured sparsity, it is possible to learn more with fewer data points. Generally, it is a good idea to search around $k_{opt}$, and since our method can achieve better performance for most values of $k_1$, rough tuning (say choose the step size to be 5, as we did on Scene, Corel and PIE, or even 10, as we did on Caltech 256) of this parameter is sufficient to get a good performance. After we set the search range and step size for $k_1$, we need to experiment with different values of $k_2$ for each $k_1$ within this range. Our method requires that $k_2 \le k_1$, and to keep our method computationally efficient, $k_2$ should not be too large (this is because the KFL part of the new regularization term has $\Theta(k_2^2)$ terms, and a large $k_2$ will results in some huge matrices in the sparse representation step, and will greatly slow down our process). From our experimental results, search $k_2 \le 30$ with a step size of 5 is sufficient to get a good performance in most cases. Note that a special case is the PIE face dataset, on which the optimal value of $k_1$ is 10. This $k_1$ is very small, which makes our method quite sensitive to the value of $k_2$. As a result, in this case, we choose the step size to be 1, and the optimal value of $k_2$ is 3. But in this case, because of the requirement $k_2 \le k_1$, the search space for $k_2$ is very small, and since $k_1$ and $k_2$ are both small, the sparse representation step is very fast, which means the search for optimal parameter values is still quite efficient.

## 7. Conclusion

In this paper, we have proposed a new graph construction method based on the $L_1$-graph and the recent development in sparse representation. Our main motivation is to overcome the traditional $L_1$-graph's potential tendency to ignore the intrinsic structure of the data and help it convey more valuable information in order to improve its performance. We have incorporated the structured sparsity into our $L_1$-graph construction, and more notably, the successful employment of $k$-NN method and kernel method enables us to deal with large scale high-dimensional datasets. The experimental results on several large complex image datasets have demonstrated that the obtained new $k$-NN fused Lasso graph consistently outperforms both the traditional $k$-NN graph and the original $L_1$-graph in the task of spectral clustering. For future work, we will apply our new $k$-NN fused Lasso graph to other graph-based machine learning tasks. Moreover, we will also make attempt to utilize other types of structured sparsity for $L_1$-graph construction.

## Acknowledgments

This work was supported by National Natural Science Foundation of China under Grants 61073084 and 61202231, Beijing Natural Science Foundation of China under Grants 4122035 and 4132037, Ph.D. Programs Foundation of Ministry of Education of China under Grant 20120001110097, and National Hi-Tech Research and Development Program (863 Program) of China under Grant 2012AA012503.

## References

[1] J. Shi, J. Malik, Normalized cuts and image segmentation, in: Proceedings of CVPR, 1997, pp. 731–737.
[2] Z. Lu, H. Ip, Constrained spectral clustering via exhaustive and efficient constraint propagation, in: Proceedings of ECCV, vol. 6, 2010, pp. 1–14.
[3] D. Zhou, O. Bousquet, T. Lal, J. Weston, B. Schölkopf, Learning with local and global consistency, in: Advances in Neural Information Processing Systems, vol. 16, 2004, pp. 321–328.
[4] X. Zhu, Z. Ghahramani, J. Lafferty, Semi-supervised learning using Gaussian fields and harmonic functions, in: Proceedings of ICML, 2003, pp. 912–919.
[5] S. Yan, D. Xu, B. Zhang, H. Zhang, Q. Yang, S. Lin, Graph embedding and extensions: a general framework for dimensionality reduction, IEEE Trans. Pattern Anal. Mach. Intell. 29 (1) (2007) 40–51.
[6] X. Zhu, Semi-supervised learning literature survey, 2006.
[7] P.P. Talukdar, Topics in Graph Construction for Semi-Supervised Learning, Technical Report, 2009.
[8] J. Wright, A. Yang, A. Ganesh, S. Sastry, Y. Ma, Robust face recognition via sparse representation, IEEE Trans. Pattern Anal. Mach. Intell. 31 (2) (2009) 210–227.
[9] S. Yan, H. Wang, Semi-supervised learning by sparse representation, in: Proceedings of SDM, 2009, pp. 792–801.
[10] B. Cheng, J. Yang, S. Yan, Y. Fu, T. Huang, Learning with l1-graph for image analysis, IEEE Trans. Image Process. 19 (4) (2010) 858–866.
[11] R. Tibshirani, Regression shrinkage and selection via the Lasso, J. R. Stat. Soc. Ser. B 58 (1) (1996) 267–288.
[12] H. Zou, T. Hastie, Regularization and variable selection via the elastic net, J. R. Stat. Soc. Ser. B 67 (2) (2005) 301–320.
[13] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, K. Knight, Sparsity and smoothness via the fused Lasso, J. R. Stat. Soc. Ser. B 67 (1) (2005) 91–108.
[14] M. Yuan, Y. Lin, Model selection and estimation in regression with grouped variables, J. R. Stat. Soc. Ser. B 68 (1) (2006) 49–67.
[15] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, New York, NY, USA, 2004.
[16] A. Oliva, A. Torralba, Modeling the shape of the scene: a holistic representation of the spatial envelope, Int. J. Comput. Vision 42 (3) (2001) 145–175.
[17] Z. Lu, H.H.-S. Ip, Spatial Markov kernels for image categorization and annotation, IEEE Trans. Syst. Man Cybern. Part B 41 (4) (2011) 976–989.
[18] X. Zheng, D. Cai, X. He, W.Y. Ma, X. Lin, Locality preserving clustering for image database, in: Proceedings of the ACM Conference on Multimedia, ACM Press, 2004, pp. 885–891.
[19] A. Yang, A. Ganesh, S. Sastry, Y. Ma, Fast L1-Minimization Algorithms and An Application in Robust Face Recognition: A Review, Technical Report No. UCB/EECS-2010-13, February 5, 2010.
[20] A. Asuncion, D.J. Newman, UCI Machine Learning Repository [http://www.ics.uci.edu/~mlearn/MLRepository.html], Irvine, CA, University of California, Department of Information and Computer Science, 2007.
[21] G. Griffin, A. Holub, P. Perona, Caltech-256 Object Category Dataset, Caltech Technical Report, 2007.
[22] Sam Roweis, Lawrence Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.
[23] T. Zhang, D. Tao, D. Zhao, X. Li, J. Yang, Patch alignment for dimension reduction, IEEE Trans. Knowl. Data Eng. 2008.
[24] Wei Bian, Dacheng Tao, Max-Min distance analysis by using sequential SDP relaxation for dimension reduction, IEEE Trans. Pattern Anal. Mach. Intell. 33 May (5) (2011) 1037–1050.
[25] N. Guan, D. Tao, Z. Luo, J. Shawe-Taylor, MahNMF: manhattan non-negative matrix factorization, CoRR abs/1207.3438, 2012.
[26] Zhou Tianyi, Tao Dacheng, Wu Xindong, Manifold elastic net: a unified framework for sparse dimension reduction, Data Min. Knowl. Discov. 22 May (3) (2011).

**Zhiwu Lu** He received the M.Sc. degree in applied mathematics from Peking University, Beijing, China in 2005, and the Ph.D. degree in computer science from City University of Hong Kong in 2011. Since March 2011, he has become an assistant professor with the Institute of Computer Science and Technology, Peking University. He has published over 30 papers in international journals and conference proceedings including TIP, TSMC-B, TMM, AAAI, ICCV, CVPR, ECCV, and ACM-MM. His research interests lie in machine learning, computer vision, and multimedia information retrieval.

**Yuxin Peng** He is the professor and director of Multimedia Information Processing Lab (MIPL) in the Institute of Computer Science and Technology (ICST), Peking University. He received his Ph.D. degree in computer application from School of Electronics Engineering and Computer Science (EECS), Peking University, in July 2003. After that, he worked as an assistant professor in ICST, Peking University. From August 2003 to November 2004, he was a visiting scholar with the Department of Computer Science, City University of Hong Kong. He was promoted to associate professor in Peking University in August 2005. In 2006, he was authorized by the "Program for New Star in Science and Technology of Beijing", and the "Program for New Century Excellent Talents in University (NCET)". In August 2010, he was promoted to professor in Peking University. He has published over 50 papers in international journals and conference proceedings including TCSVT, TIP, ACM-MM, ICCV, CVPR and AAAI. In 2009, he led his team to participate in TRECVID. In six tasks of the high-level feature extraction (HLFE) and search, his team won the first places in fours tasks and the second places in the left two tasks. Besides, he has obtained 12 patents. His current research interests include multimedia information retrieval, computer vision and pattern recognition.