Int J Data Sci Anal
DOI 10.1007/s41060-016-0020-3

REGULAR PAPER

A greedy algorithm to construct sparse graph by using ranked dictionary

Shuchu Han[1] · Hong Qin[1]

Received: 14 July 2016 / Accepted: 3 August 2016

Abstract $\mathcal{L}_1$ graph is an effective way to represent data samples in many graph-oriented machine learning applications. Its original construction algorithm is nonparametric, and the graphs it generates may have high sparsity. Meanwhile, the construction algorithm also requires many iterative convex optimization calculations and is very time-consuming. Such characteristics would severely limit the application scope of $\mathcal{L}_1$ graph in many real-world tasks. In this paper, we design a greedy algorithm to speed up the construction of $\mathcal{L}_1$ graph. Moreover, we introduce the concept of "Ranked Dictionary" for $\mathcal{L}_1$ minimization. This ranked dictionary not only preserves the locality but also removes the randomness of neighborhood selection during the process of graph construction. To demonstrate the effectiveness of our proposed algorithm, we present our experimental results on several commonly used datasets using two different ranking strategies: One is based on Euclidean distance, and another is based on diffusion distance.

Keywords Graph based learning · Sparse graph · Spectral clustering · Subspace learning

This paper is an extension version of the PAKDD2016 Long Presentation paper: "A Greedy Algorithm to Construct L1 Graph with Ranked Dictionary".

✉ Shuchu Han
shhan@cs.stonybrook.edu

Hong Qin
qin@cs.stonybrook.edu

1 Stony Brook University, Stony Brook, NY, USA

1 Introduction

For graph-oriented learning tasks, a quality graph representation [6] of input data samples is the key to success. In the past few decades, researchers in machine learning area propose many different methods to solve such tasks, for example, k nearest neighbor (kNN) graph and ϵ-ball graphs. These methods are very straightforward and proved to be efficient for general data. The reason of these methods' success is that their construction algorithm acts as a local smooth "filter" which sets the weight between faraway data points and source point to zero. The built graph is constructed by many such local star-shaped patches (or subgraphs). However, both of them need a user-specified parameter such as k or ϵ which is chosen empirically. Considering the versatility and uncertainty of the real-world data, a bad selection of parameter k and ϵ will lead to an inaccurate conclusion for subsequent machine learning tasks. Recently, a nonparametric graph called $\mathcal{L}_1$ graph is proposed by Cheng et al. [4]. Based on existing sparse representation frameworks [21,25], the construction algorithm of $\mathcal{L}_1$ graph can be described as follows: Given an input data samples $\mathbf{X} = [\mathbf{x_1}, \mathbf{x_2}, \dots, \mathbf{x_n}]$, where each $\boldsymbol{x_i}, i \in [1, \dots, n]$ is a vector that represents one single data sample. The $\mathcal{L}_1$ graph of $\mathbf{X}$ is built by finding a sparse coding [23] of each $\mathbf{x_i}$ with a dictionary constructed from all data samples except $\mathbf{x_i}$ itself. The coefficient of sparse coding is used as the edge weight of resulted $\mathcal{L}_1$-graph. The mathematical definition of sparse coding is:

$$\textbf{(P1)} \quad \min_{\boldsymbol{\alpha_i}} \|\boldsymbol{\alpha_i}\|_1 \text{ subject to } \boldsymbol{x_i} = \boldsymbol{\Phi}^i \boldsymbol{\alpha_i}, \tag{1}$$

where dictionary $\boldsymbol{\Phi}^i = [\boldsymbol{x_1}, \dots, \boldsymbol{x_{i-1}}, \boldsymbol{x_{i+1}}, \dots, \boldsymbol{x_n}]$, and $\boldsymbol{\alpha}_i \in \mathbb{R}^{n-1}$ is the sparse code of $\boldsymbol{x_i}$. The coefficients of $\boldsymbol{\alpha}_i$ could be negative, depending on the choices of $\mathcal{L}_1$ minimization solvers. To make them have the physical meaning of "Similarity," the absolute value or nonnegative constraints are employed.

As we could see from the above description, the $\mathcal{L}_1$ graph construction algorithm is nonparametric and the user is not required to input any parameters except for the solver. The construction algorithm is a pure numerical process based on convex optimization. Cheng et al. [4] show that $\mathcal{L}_1$ graph has three advantages comparing traditional graph construction methods. They are: (1) robustness to data noise; (2) sparsity; (3) datum-adaptive neighborhood. Their experimental results also prove that $\mathcal{L}_1$ graph has significant performance improvement in many machine learning applications such as spectral clustering, subspace learning and semi-supervised learning [4]. Nevertheless, just like each sword has double edges, $\mathcal{L}_1$ graph also bears some disadvantages such as: (1) sensitive to duplications. For example, if every data sample has a duplication, the resulted $\mathcal{L}_1$ graph will only have edge connections between the data sample and its duplication; (2) randomness, the edge and edge weight are highly dependent on the solver; (3) high computational cost [4]; (4) lost of the locality [10,11,32]. To overcome these disadvantages, many improved algorithms have been proposed in recent years. Now, we would like to classify them into two categories: *soft-modification* and *hard-modification*.

1. *Soft-modification* algorithms. Algorithms in this category usually add one or more regularization terms to the original $\mathcal{L}_1$ minimization objective function in Eq. (1). For example, the structure sparsity [33] preserves the local structure information of input data, the auto-grouped sparse regularization [10] adds the group effect to the final graph, and the graph Laplacian regularization [29,30] lets the closed data samples have similar sparse coding coefficients (or $\boldsymbol{\alpha}_i$).
2. *Hard-modification* algorithms. These algorithms define a new dictionary for each data sample during $\mathcal{L}_1$ minimization. By reducing the solvers' solution space for each data sample into a local space, the locality of input data is preserved and the computational time of $\mathcal{L}_1$ minimization [Eq. (1)] is reduced. For example, the locality preserved (LOP) $\mathcal{L}_1$ graph is utilizing k nearest neighbors as dictionaries [11].

The *soft-modification* algorithms preserve the nonparametric feature and improve the quality of $\mathcal{L}_1$ graph by exploiting the intrinsic data information such as geometry structure and group effects. However, those algorithms still have high computational cost. This is unpleasant for the large-scale dataset in this "Big-data" era. To improve, in this paper we propose a greedy algorithm to generate $\mathcal{L}_1$ graph. The generated graphs are called **Greedy-$\mathcal{L}_1$** graphs. Our algorithm employs greedy $\mathcal{L}_1$ minimization solvers and is based on nonnegative orthogonal matching pursuit (NNOMP). Furthermore, we use ranked dictionaries with reduced size K which is a user-specified parameter. We provide the freedom to the user to determine the ranking strategy such as nearest neighbors or diffusion ranking [5]. Our algorithm has significant time reduction about generating $\mathcal{L}_1$ graphs. Comparing the original $\mathcal{L}_1$ graph construction method, our algorithm loses the nonparametric characteristics and is only offering a suboptimal solution. However, our experimental results show that the graph generated by our algorithm has equal (or even better) performance as the original $\mathcal{L}_1$ graph by setting K equals to the length of data sample. Our work is a natural extension of existing $\mathcal{L}_1$ graph research. A concise summary of the connection between our proposed Greedy-$\mathcal{L}_1$ graph and other graphs is illustrated in Figs. 1 and 2. This paper is an improved version of accepted conference paper [12]. We summary the main contributions as:

1. We propose a greedy algorithm to reduce the computational time of generating $\mathcal{L}_1$ graph.
2. We introduce the ranked dictionary for $\mathcal{L}_1$ minimization solver. This new dictionary not only reduces the time of minimization process but also preserves the locality and geometry structure information of input data.
3. Our algorithm removes the randomness of edges in final $\mathcal{L}_1$ graph and preserves the uniqueness except for the edge weights. Moreover, our algorithm can generate $\mathcal{L}_1$ graphs with lower sparsity.
4. We present experiment and analysis results by applying our algorithm to spectral clustering application with different datasets. Our experimental results show that the graphs generated by our proposed greedy algorithm have equal clustering performance even though it is only providing a suboptimal solution.
5. We discuss the connection between $\mathcal{L}_1$ graph and sparse subspace clustering and also explain the different with locally linear embedding [19].

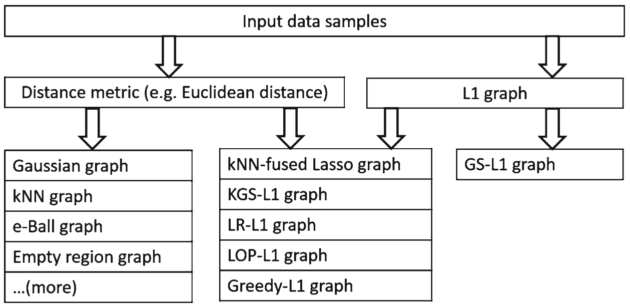

Fig. 1 Connection of greedy $\mathcal{L}_1$ graph to other graphs. Several of them are: kNN-fused Lasso graph [33], group sparse (GS) $\mathcal{L}_1$ graph, kernelized group sparse (KGS) $\mathcal{L}_1$ graph [10], Laplacian regularized (LR) $\mathcal{L}_1$ graph [30] and locality preserving (LOP) $\mathcal{L}_1$ graph [11]

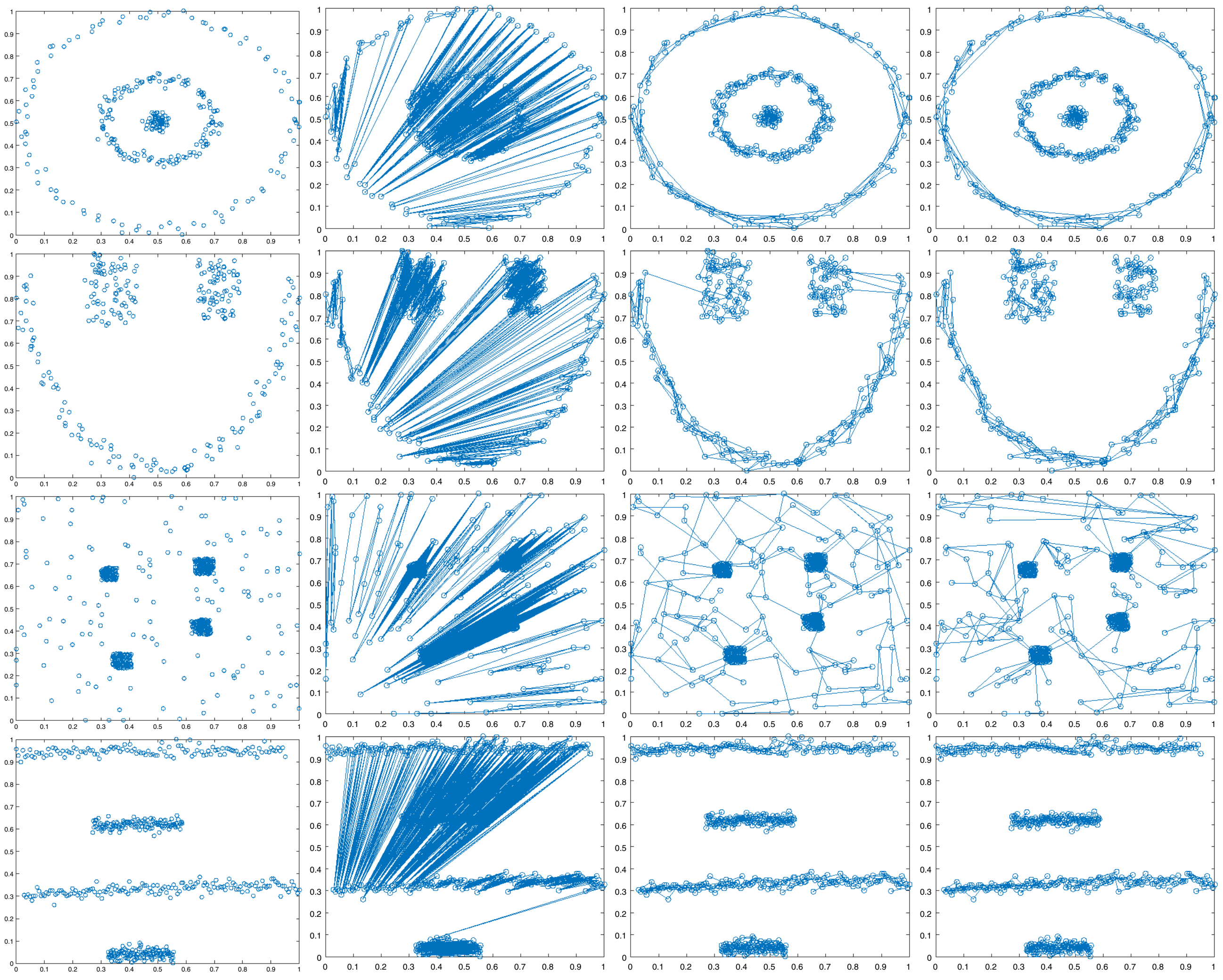

Fig. 2 $\mathcal{L}_1$ graphs generated by different construction algorithms. From left to right: 2D toy dataset; $\mathcal{L}_1$ graph; Greedy-$\mathcal{L}_1$ graph with Euclidean distance ranking ($K = 15$); Greedy-$\mathcal{L}_1$ graph with diffusion distance ranking ($K = 15$)

6. We review the recent research works related to $\mathcal{L}_1$ graph such as $\mathcal{L}_0$ graph which has better subspace persevering ability in theory [28] and scalable sparse subspace clustering [31] for the computational challenges of large size data.

The organization of our paper is as follows. First, an overview of the disadvantages of original $\mathcal{L}_1$ graph construction algorithm will be presented in Sect. 2. Second, we will introduce our proposed greedy algorithm in Sect. 3. After that, we will give a review of existing works on how to improve the quality of $\mathcal{L}_1$ graph. Finally, we will present our experimental results in Sect. 5 and draw conclusion in Sect. 6.

2 Overview

In this section, we make our attempts to address two problems of original $\mathcal{L}_1$ graph construction algorithm. They are: (1) curse of dictionary normalization and (2) nonlocal edges.

2.1 Original $\mathcal{L}_1$ graph construction algorithm

This section reviews the original construction algorithm of $\mathcal{L}_1$ graph (1) in advance for the purpose of comparison with our proposed new algorithm in Sect. 3. The details can be seen as algorithm (1). As we can see from the original construction algorithm, it includes n repeat optimization steps, while n equals to the number of data samples. Moreover, each optimization step involves an over-complete dictionary Φ^i which

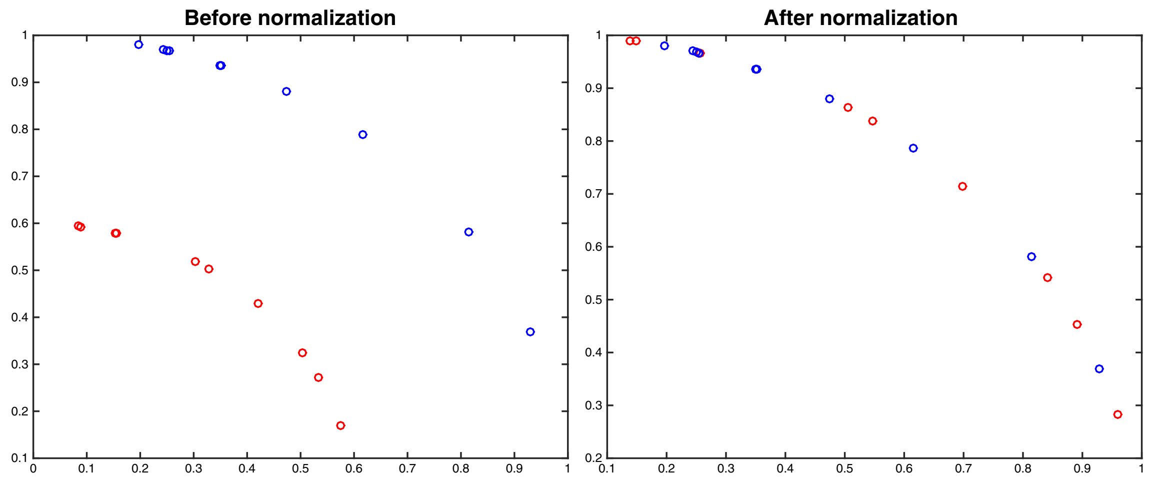

Fig. 3 Demonstration of dictionary normalization for a toy dataset. The *red* (bottom left) and *blue* (up right) points represent two different clusters. *Left* before normalization; *right* after normalization. We can see that the neighborhood relationship is changed after normalization (color figure online)

has size $n-1$. These properties label the original construction algorithm with "time-consuming." Nevertheless, the original construction algorithm also bears two other disadvantages as we will go details in following two sections.

Algorithm 1: $\mathcal{L}_1$-graph [4].

```
Input  : Data samples $\mathbf{X} = [\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}]$, where $\mathbf{x_i} \in \mathbb{R}^d$;
Output: Adjacency matrix $\mathbf{W}$.
1  Normalize each data sample $\mathbf{x_i}$ to has unit length: $\|\mathbf{x_i}\|_2 = 1$;
2  for $\mathbf{x_i} \in \mathbf{X}$ do
3      $\mathbf{\Phi^i} = [x_1, x_2, x_{i-1}, \ldots, x_{i+1}, x_n]$;
4      Solve the following optimization problem:
       $\min_{\alpha_i} \|\alpha_\mathbf{i}\|_1, \quad s.t. \quad \mathbf{x_i} = \mathbf{\Phi^i}\alpha_\mathbf{i}$;
5      if $j < i$ then
6          $W_{ij} = \alpha_\mathbf{i}(j)$
7      else if $j \geq i$ then
8          $W_{ij} = \alpha_\mathbf{i}(j-1)$
9      end
10     $W_{ii} = 0$;
11 end
   /* Modify $W$ into a similarity matrix
      manually.                                  */
12 $W = (|W| + |W'|)/2$;
13 Return $\mathbf{W}$.
```

2.2 Curse of dictionary normalization

While solving $\mathcal{L}_1$ minimization, the atoms of dictionary are normalized to have unit length. The goal of this step is to satisfy the theoretic requirement of compressive sensing. The less-ideal part about this normalization is that it does not preserve neighborhood information of input data. This can be illustrated in Fig. 3. To illustrate this phenomenon, we manually create a toy dataset in 2D and it has two clusters visually. After normalization, we can see that the neighbors of a node are changed. This normalization step projects all data samples onto a unit hypersphere, and the original geometry structure information is lost.

2.3 Nonlocal edges

During the construction of $\mathcal{L}_1$ graph, an over-complete dictionary is created for each data sample. The original method simply selects all other data samples as the dictionary. This strategy affords the *nonparametric* property of $\mathcal{L}_1$ graph. However, it also introduces nonlocal edges. In other words, it does not preserve the *locality* of input data [11]. This phenomenon can be illustrated in Fig. 4.

2.4 Unstable solutions caused by different $\mathcal{L}_1$ solvers

To solve the optimization problem (1), we need a numerical solver. There are many popular ones with various minimization methods [27] such as*gradient projection, homotopy* and *proximal gradient*. Moreover, all these solvers have their own special parameter settings. As a result, if we choose different parameters, the numerical results will be not same, or say, unstable. To illustrate this phenomenon, we exam the UCI Image dataset with "spams-matlab" software [16] and "l1_ls" software [13]. For each solver, we set the parameter λ to different values as: $[0.01, 0.001, 0.0001]$. For the experiment, we select the first sample of image dataset as source sample and others as dictionary. To see the unstable solutions, we list the top five neighbors (Index) and its corresponding weights (Value). The result is shown in Table 1. As we can see, the majority neighbors between "spams-matlab" and "l1_ls" are same except some minor difference. However,

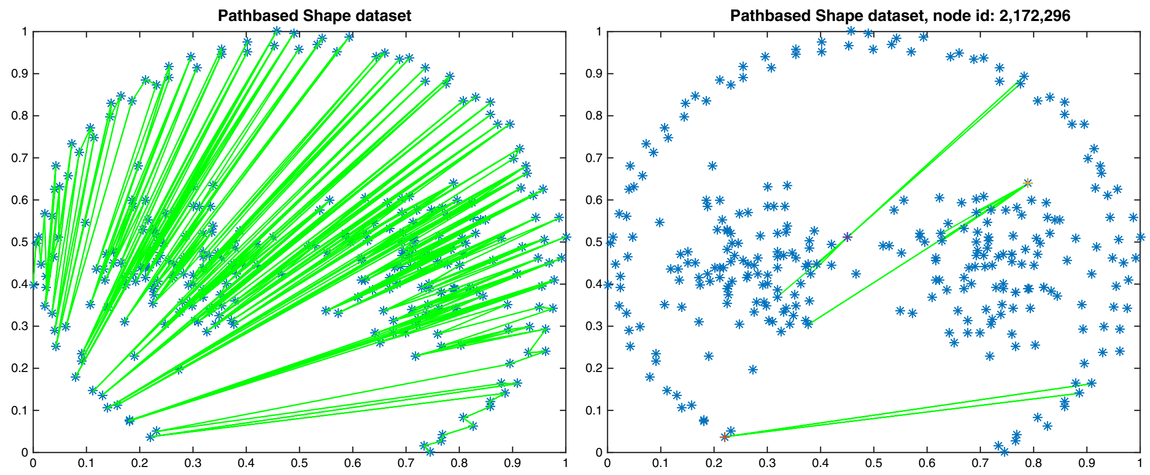

Fig. 4 $\mathcal{L}_1$-graph of path-based dataset. *Left* the entire graph; *right* edges of three selected points. We can see the existence of nonlocal edges

Table 1 The effect of unstable solutions caused by using different solvers or with different parameters

Solver	λ	Index (value)
L1-ls	0.01	5 (0.2111), 14 (0.4449), 17 (0.2718), 38 (0.0418), 575 (0.0163)
Spams-matlab	0.01	5 (0.2632), 13 (0.0044), 14 (0.3525), 17 (0.2819)
L1-ls	0.001	5 (0.0771), 14 (0.4540), 17 (0.3005), 38 (0.0715), 575 (0.0908)
Spams-matlab	0.001	5 (0.2851), 14 (0.3676), 17 (0.3142), 38 (0.0043)
L1-ls	0.0001	14 (0.3861), 17 (0.4051), 32 (0.0292), 36 (0.0211), 575 (0.1413)
Spams-matlab	0.0001	5 (0.2621), 14 (0.4171), 17 (0.2744), 38 (0.0346), 225 (0.0068)

the weights are very different and unstable. This unstable situation is not only with different parameter λ, but also with different solvers. This is a disadvantage for using $\mathcal{L}_1$ graph as similarity graph for graph-oriented machine learning tasks.

3 New algorithm

In this section, we introduce the concept of ranked dictionary and two different ranking strategies: Euclidean distance ranking and diffusion ranking. These different ranking methods are proposed for different types of data. For example, diffusion ranking is suitable for data with manifold structure, and Euclidean distance is the popular choice for general data. Obviously, there are many other distance choices such as cosine distance could be used for ranking, and it is upon user's judgment for the right choice. Furthermore, we present a greedy algorithm at the end of this section.

3.1 Ranked dictionary

We propose a "ranked dictionary" to substitute the original dictionary Φ^i in Eq. (1). Our claim is that the "ranked dictionary" not only preserves the locality of data, which is important for clustering applications, but also solves the "curse of dictionary normalization" dilemma. The idea of "ranked dictionary" is to rank the neighborhood information following a given distance metric such as Euclidean distance in vector space. By selecting the top K nearest neighbors as dictionary, the new dictionary Φ^i_K keeps the order of nearest neighbors and captures the local distribution of data samples. Moreover, Φ^i_K has smaller size comparing $n-1$, while n equals to the number of data samples.

There is a subtle difference between k value of popular k nearest neighbor (kNN) graph and the K value in our proposed "ranked dictionary" Φ^i_K. Usually, the users set the value k of kNN graph in the order of $\log(n)$ which is the asymptotic connectivity result [1] that makes the kNN graph to be connected. For K value of Φ^i_K, it needs to be larger than d which is the dimension of vector x_i. This requirement is to let Φ^i_K be an over-complete dictionary.

The use of truncated version of dictionary Φ is proved to be success in building quality $\mathcal{L}_1$ graph for clustering application [11]. However, it cannot solve the dilemma that there might exist data samples with the same direction but different lengths in input data. The dictionary normalization process will project them onto the same location at hypersphere. Since they have the same values, the $\mathcal{L}_1$ minimization

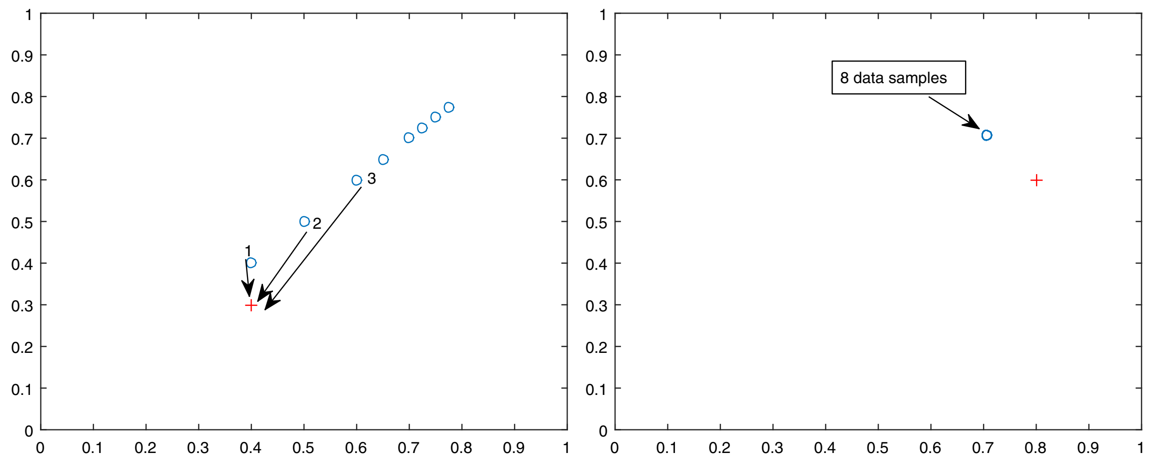

Fig. 5 Ranked dictionary. *Left* eight data samples have the same direction but with different lengths. *Red cross* is the target data sample for calculating sparse coefficients. *Right* after normalization, those eight data samples have the same location (color figure online)

solver will choose one of them randomly. To avoid this randomness, we need to rank those atoms (or data samples) of dictionary (Fig. 5).

Euclidean Distance Ranking. Using Euclidean metric to rank atoms of dictionary is quite straightforward. We rank them by distance. The shorter distance will have a higher rank score. The Euclidean distance is defined as:

$$\text{dist}\,(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 = \left(\sum_{k=1}^{n} |x_i(k) - x_j(k)|^2\right)^{1/2}. \tag{2}$$

Diffusion Distance Ranking. As pointed out by Yang et al. [30], many real-world datasets are similar to an intrinsic low-dimensional manifold embedded in high-dimensional ambient space, and the geometry structure of manifold can be used to improve the performance of learning algorithms. We now present a strategy to search dictionaries following the geometry structure of input data. Based on the diffusion theory [5,7], we rank the atoms of dictionary through diffusion matrix. A diffusion process has three stages [7]: (1) initialization; (2) definition of transition matrix; (3) definition of the diffusion process. In our setting, the first stage is to build an affinity matrix $\boldsymbol{A}$ from the input dataset $\boldsymbol{X}$. We use Gaussian kernel to define the pairwise distance:

$$\boldsymbol{A}(i, j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right), \tag{3}$$

where $\boldsymbol{A}(i, j)$ is the distance between data sample $\boldsymbol{x}_i$ and data sample $\boldsymbol{x}_j$, and σ is a normalization parameter. In our configuration, we use the median of K nearest neighbors to tune σ. The second stage is to define the transition matrix $\boldsymbol{P}$:

$$\boldsymbol{P} = \boldsymbol{D}^{-1}\boldsymbol{A}, \tag{4}$$

where $\boldsymbol{D}$ is a $n \times n$ degree matrix defined as

$$\boldsymbol{D}(i, j) = \begin{cases} \sum_{j=1}^{n} \boldsymbol{A}(i, j) & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Now the diffusion process can be defined as:

$$\boldsymbol{W}_{t+1} = \boldsymbol{P}\boldsymbol{W}_t\boldsymbol{P}', \tag{6}$$

where $\boldsymbol{W}_0 = \boldsymbol{A}$ and t is the number of steps for diffusion steps. Each row of $\boldsymbol{W}_t$ is the diffusion ranking scores. In this paper, we let t equal to K for the sake of simplicity. Once $\boldsymbol{W}_t$ is calculated, the first K data samples with top scores of each row are selected as dictionary. The algorithmic details can be documented as follows:

3.2 Greedy $\mathcal{L}_1$ graph

To solve the $\mathcal{L}_1$ norm minimization problem, we need an efficient solver [27]. For datasets that size are larger than 3,000 with reasonable dimensions, greedy solver like Basic Pursuit (BP) [3,8] or Orthogonal Matching Pursuit (OMP) [22] is more suitable [31]. In this section, We propose a greedy algorithm to build $\mathcal{L}_1$ graph. Our proposed algorithm is based on OMP [31] and NNOMP [2,15]. By using greedy solver, we switch the $\mathcal{L}_1$ minimization problem (**P1**) back to the original $\mathcal{L}_0$ optimization with (**P2**)/without (**P3**) nonnegative constraints as:

$$(\mathbf{P2}) \quad \min_{\boldsymbol{\alpha}_i} \|\boldsymbol{\alpha}_i\|_0 \text{ subject to } \boldsymbol{x}_i = \boldsymbol{\Phi}^i\boldsymbol{\alpha}_i, \boldsymbol{\alpha}_i \geq 0. \tag{7}$$

$$(\mathbf{P3}) \quad \min_{\boldsymbol{\alpha}_i} \|\boldsymbol{\alpha}_i\|_0 \text{ subject to } \boldsymbol{x}_i = \boldsymbol{\Phi}^i\boldsymbol{\alpha}_i. \tag{8}$$

Algorithm 2: DiffusionDictionary

```
Input  : Data samples $X = [x_1, x_2, ..., x_n]$, where $x_i \in X$;
         Size of dictionary: $K$;
Output: Diffusion dictionary index matrix $\Phi_K$.
1  Calculate Gaussian similarity graph $A$;
2  $P = D^{-1}A$;
   /* calculate diffusion process iteratively.
      */
3  for $t = 1 : K$ do
4      $W_t = PW_{t-1}P'$
5  end
   /* sort each row in descend order.          */
6  for $i = 1 : n$ do
7      sort($W_t(i, :)$)
8  end
   /* fetch the index of highest $K$ values in
      each row of $W_t$                          */
9  for $i = 1 : n$ do
10     $\Phi(i, :)$ =index($W_t(i, 1 : k)$)
11 end
```

The main difference between our algorithm and the original OMP and NNOMP is that the atoms of dictionary are ranked. We force the solver to choose and assign weights to atoms that are closer to source data sample before normalization. To clarify our idea, we present the improved version of NNOMP solver in algorithm (3). For OMP solver, the idea and process are same.

3.3 Connection to subspace clustering

$\mathcal{L}_1$ graph is almost the same as the similarity graph of sparse subspace clustering (SSC) algorithm [9]. However, they have different assumptions about the data. The $\mathcal{L}_1$ graph is defined for general data and does not have any specific assumption about data like k nearest neighbor graph, while the similarity graph of SSC assumes the data are lied in a union of low-dimensional subspaces [9].

The success of $\mathcal{L}_1$ graph is first applied to human face images clustering [25,26]. Those face images data have two sufficient conditions for the success of using $\mathcal{L}_1$ graph for spectral clustering: (1) The dimension of data vector is high. (2) Different human face images stay in different subspaces. However, for general data, these two conditions not always exist. By the experimental results from research work [11], the Ng, Jordan, Weiss et al. (NJW) spectral clustering algorithm [18] with Gaussian similarity graph has better performance than with $\mathcal{L}_1$ graph on several general datasets. So here, we argue that the power of $\mathcal{L}_1$ graph follows the assumption of sparse subspace clustering.

3.4 Connection to locally linear embedding

The idea of "ranked dictionary" has a connection to locally linear embedding (LLE) [19]. LLE solves the following minimization problem:

Algorithm 3: Greedy$\mathcal{L}_1$Graph.

```
Input  : Data sample $x$;
         Ranked dictionary $\Phi_K$;
         Residual threshold $\theta_{threshold}$
Output: Sparse coding $\alpha$ of $x$.
1  for $i = 1 : \|x\|_1$ do
2      if $i == 0$ then
3          Temporary solution: $\alpha^i = 0$;
4          Temporary residual: $r^i = x - \Phi_K \alpha^i$;
5          Temporary solution support: $S^i = Support\{\alpha^i\} = \emptyset$;
6      else
7          for $j = 1 : k$ do
               /* $\phi_j$ is the $j$-th atom of $\Phi_K$        */
8              $\epsilon(j) = \min_{\alpha_j \geq 0} \|\phi_j \alpha_j - r^{i-1}\|_2^2 =$
               $\|r^{i-1}\|_2^2 - \max\{\phi_j^T r^{i-1}, 0\}^2$.
9          end
10         Find $j_0$ such that $\forall j \in S^c, \epsilon(j_0) \leq \epsilon(j)$, if there are
           multiple $j_0$ atoms, choose the one with smallest index
           value.;
11         Update support: $S^i = S^{i-1} \cup \{j_0\}$;
12         Update solution: $\alpha^i = \min_z \|\Phi_K \alpha - x\|_2^2$ subject to
           $Support\{\alpha^i\} = S^i$ and $\alpha^i \geq 0$;
13         Update residual: $r^i = x - \Phi_K \alpha^i$;
14         if $\|r^i\|_2^2 < \theta_{threshold}$ then
15             Break;
16         end
17     end
18 end
19 Return $\alpha^i$;
```

$$\epsilon(w) = \sum_i |\mathbf{x}_i - \sum_j \mathbf{w}_{ij}\mathbf{x}_j|^2. \tag{9}$$

The cost function $\epsilon(w)$ is the add up of the squared distance between all data samples (x_i) and their reconstructions $\sum_j \mathbf{w}_{ij}\mathbf{x}_j$. There are two constraints during the minimization process: (1) The $\mathbf{x}_j$ are selected to be k nearest neighbor of $\mathbf{x}_i$, where k is a parameter set by user; (2) the row of weight matrix sum to one: $\sum_j \mathbf{w}_{ij} = 1$.

If we compare Eq. 9 of LLE with Eq. 1 of $\mathcal{L}_1$ graph and "ranked dictionary," we can find that both of them are finding a linear representation relationship between a given data sample and its k nearest neighbors. However, $\mathcal{L}_1$ graph with "ranked dictionary" looks for a sparse reconstruction weights and prefer to assign nonzero weights for nearest neighbors $\mathbf{x}_j$ that stay in the same subspace as the given data sample $\mathbf{x}_i$. The second difference is the unique advantage of $\mathcal{L}_1$ graph.

3.5 Quality as similarity graph for spectral clustering

One major application of $\mathcal{L}_1$ graph is spectral clustering. Researchers use $\mathcal{L}_1$ graph as the similarity graph of spectral clustering algorithm by treating the sparse coefficients

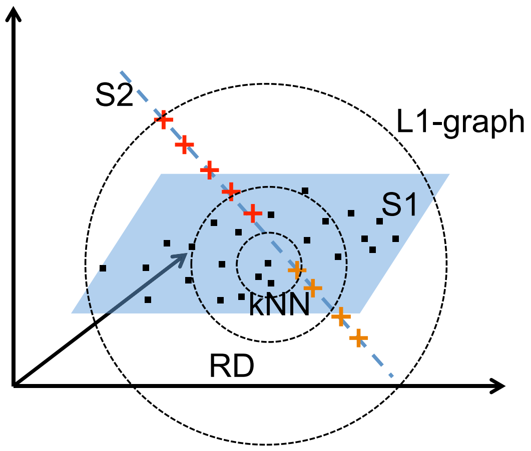

Fig. 6 The range difference of "Ranked Dictionary" (RD), "kNN" and original "$\mathcal{L}_1$ graph." The toy dataset includes two subspace *S1* and *S2*. The selection range of nearest neighbors is shown by *dash circles*

as similarity values. The similarity graph models the cluster structure of original data with pre-defined similarity metric and has significant impact on the performance of spectral clustering algorithm. A good similarity graph should have high weights for edges within same cluster and low weights for edges between different clusters. However, there is no explicit measurement of the quality of similarity graph from theoretic research as pointed out by [24]. Instead, the clustering performance, like Mutual Information and Accuracy, is used to tell whether the similarity graph is in high quality or not implicitly. "Locality" is another guidance to judge the quality of similarity graph [10]. "Locality" stresses that the edges of similarity graph should connect data points locally as nonlocal edges will affect the result of graph cut [20] and then the performance of spectral clustering [24]. In this section, we try to explain how $\mathcal{L}_1$ graph with "ranked dictionary" can generate high-quality similarity graphs.

"Ranked dictionary" preserves the locality of data by only selecting k nearest neighbors as dictionary. For a given source data point, "ranked dictionary" constrains the possible candidates that it can connect to. There is a difference between k nearest neighbor of kNN graph and our proposed greedy $\mathcal{L}_1$ graph. We show it in Fig. 6.

As we can see, greedy $\mathcal{L}_1$ graph selects a larger range than kNN graph but a much smaller one than original $\mathcal{L}_1$ graph. It preserves the locality of data in a "Hard-modification" way as we introduced in the beginning of this work. And this locality preserving ability has been proved in previous research work [11].

Another important aspect of greedy $\mathcal{L}_1$ graph is that it preserves the local subspaces through OMP solver. As the theory proof in [31], if coherence between the residual vectors (set of r^i in line 13 of algorithm (3)) and subspaces satisfies a data-dependent condition, the OMP solver preserves the subspaces of input data. Based on this, we observe another difference with kNN graph: The greedy $\mathcal{L}_1$ graph prefers to create connections between data samples within same subspace, while the kNN graph selects edges according to the given distance metric.

4 Related works

Original $\mathcal{L}_1$ graph [4] is a pure numerical result and does not exploit the physical and geometric information of input data. To improve the quality of $\mathcal{L}_1$ graph, several research works are proposed to use the intrinsic structure information of data by adding one or several regularization terms to the $\mathcal{L}_1$ minimization **P1**. For example, consider the elastic net regularization [10], OSCAR regularization [10] and graph Laplacian regularization [30].

Another research direction of $\mathcal{L}_1$ graph is to reduce its high computational cost. Zhou et al. [33] propose a kNN-fused Lasso graph by using the idea of k nearest neighbors in kernel feature space. With a similar goal, Fang et al. [10] propose an algorithm which transfers the data into reproducing kernel Hilbert space and then projects them into a lower-dimensional subspace. By these operations, the dimension of the dataset is reduced and the computational time is reduced.

$\mathcal{L}_1$ graph has strong theory connection with sparse subspace clustering [9]. Recently, Yingzhen et al. [28] propose the $\mathcal{L}_0$ graph which try to solve the same minimization problem **P3** as in this paper. They intuition is that $\mathcal{L}_0$ norm has better subspace preserving ability than $\mathcal{L}_1$ norm. Since minimization $\mathcal{L}_0$ norm is NP-hard, there are more nonconvex optimization challenges remained to be solved.

5 Experiments

We present our experimental results in this section. The datasets in our experiments can be divided into small size data and large size data. The reason for this separation is that calculating the global optimization for $\mathcal{L}_1$ minimization is time-consuming for large size data (number of instances are larger than 3000.) For those large size data, we use an efficient OMP solver from "spams-matlab" [16]. As a consequence, the generated $\mathcal{L}_1$ graphs are not from optimal sparse coding solutions.

The effectiveness of our proposed graph construction methods is evaluated through NJW spectral clustering algorithm [18]. To satisfy the input of spectral clustering algorithm, we transform the adjacency matrix of $\mathcal{L}_1$ graph $\boldsymbol{W}$ into a symmetry matrix $\boldsymbol{W}'$ by $\boldsymbol{W}' = (\boldsymbol{W} + \boldsymbol{W}^T)/2$. All analyses and experiments are carried out by using MATLAB on a server with Intel 12-core 3.4GHz CPU and 64GB RAM.

Solvers We use three solvers in our experiments. For small size dataset, "l1-ls" is used for creating $\mathcal{L}_1$ graph, and our proposed NNOMP solver is used for greedy $\mathcal{L}_1$ graph. For

Table 2 Statistics of small size datasets

Name	#samples	#attributes	#clusters
BreastTissue (BT)	106	9	6
Iris	150	4	3
Wine	178	13	3
Glass	214	9	6
Soybean	307	35	19
Vehicle	846	18	4
Image	2100	19	7

large dataset, we use "spams-matlab" software [16], which is an efficient implementation of sparse optimization by using multi-thread techniques, to build the $\mathcal{L}_1$ graph and greedy $\mathcal{L}_1$ graph.

Evaluation metrics We evaluate the spectral clustering performance with Normalized Mutual Information (NMI) and Accuracy (ACC). NMI value ranges from 0 to 1, with higher values meaning better clustering performance. AC is another metric to evaluate the clustering performance by measuring the fraction of its clustering result that is correct. Its value also ranges from 0 to 1, and the higher the better.

5.1 Small size data

5.1.1 Datasets

To demonstrate the performance of our proposed algorithm, we evaluate it on seven UCI benchmark datasets including three biological datasets (BreastTissue, Iris and Soybean), two vision image datasets (Vehicle and Image), one chemistry dataset (Wine) and one physical dataset (Glass), whose statistics are summarized in Table 2. All of these datasets have been popularly used in spectral clustering analysis research. The diverse combinations of datasets are necessary for our comprehensive studies.

5.1.2 Baselines and parameters setting

We compare the spectral clustering performance with Gaussian similarity graph and original $\mathcal{L}_1$ graph. For experiments with small size datasets, we use the *l1_ls* solver [14] for original $\mathcal{L}_1$ graph construction algorithms. We set the solver's parameter λ to 0.1. The threshold $\theta_{\text{threshold}}$ of greedy solver 3 is set to $1e-5$. For Gaussian graph and greedy-$\mathcal{L}_1$ graph, we select three different K values and document their clustering performance results, respectively. The K is set to be the multiple data attribute sizes. The results are documented in Tables 3 and 4.

5.1.3 Greedy-$\mathcal{L}_1$ graph versus Gaussian graph

Overall, the greedy-$\mathcal{L}_1$ graph using Euclidean metric has better average spectral clustering performance than Gaussian graphs. However, we also observe that Gaussian graph has overall better performance on "Iris," "Soybean" and "Vehicle" datasets.

5.1.4 Greedy-$\mathcal{L}_1$ graph versus $\mathcal{L}_1$ graph

Greedy-$\mathcal{L}_1$ graph has better clustering performance than $\mathcal{L}_1$ graph on average. However, for Iris and Soybean datasets, the $\mathcal{L}_1$ graph shows the best clustering result: Iris (NMI = 0.5943, ACC = 0.74); Soybean (NMI = 0.7373, ACC = 0.6156). The best result of greedy-$\mathcal{L}_1$ graph is: Iris (NMI = 0.5106, ACC = 0.72); Soybean (NMI = 0.6919, ACC = 0.5244).

5.1.5 Euclidean distance ranking versus diffusion ranking

The Euclidean distance ranking appears to have better clustering performance than that of diffusion ranking in general. This is rather a surprising result to us. Only for "Iris" dataset,

Table 3 NMI comparison of graph construction algorithms

Name	$\mathcal{L}_1$ graph	Gaussian graph	Greedy-$\mathcal{L}_1$ graph (Euclidean)			Greedy-$\mathcal{L}_1$ graph (diffusion)		
			$K=1\times M$	$K=2\times M$	$K=3\times M$	$K=1\times M$	$K=2\times M$	$K=3\times M$
BT	0.4582	0.4606	0.5473	0.4517	0.5024	0.4197	0.4073	0.3839
Iris	0.5943	0.7364	0.3950	0.4623	0.4070	0.5106	0.4626	0.4640
Wine	0.7717	0.8002	0.8943	0.9072	0.8566	0.6925	0.4291	0.6093
Glass	0.3581	0.2997	0.2569	0.3688	0.3039	0.2991	0.3056	0.2918
Soybean	0.7373	0.6958	0.6919	0.6833	0.6775	0.5788	0.5493	0.5432
Vehicle	0.1044	0.1870	0.1512	0.2121	0.2067	0.1438	0.1035	0.1244
Image	0.4969	0.4652	0.5821	0.6673	0.6649	0.4866	0.4483	0.3155
Average	0.5030	0.5207	0.5170	**0.5361**	0.5170	0.4473	0.3865	0.3903

M is the number of attributes
Bold value indicates the best performance

Table 4 ACC comparison of different graph construction algorithms

Name	$\mathcal{L}_1$ graph	Gaussian graph	Greedy-$\mathcal{L}_1$ graph (Euclidean)			Greedy-$\mathcal{L}_1$ graph (Diffusion)		
			$K = 1 \times M$	$K = 2 \times M$	$K = 3 \times M$	$K = 1 \times M$	$K = 2 \times M$	$K = 3 \times M$
BT	0.5472	0.5377	0.6698	0.4811	0.5943	0.4528	0.4906	0.4717
Iris	0.7400	0.8867	0.6933	0.7200	0.6800	0.7200	0.6533	0.6400
Wine	0.9326	0.9438	0.9719	0.9719	0.9551	0.8989	0.7865	0.8596
Glass	0.4206	0.4112	0.4579	0.4533	0.4346	0.4626	0.4813	0.5187
Soybean	0.6156	0.5440	0.5244	0.4853	0.5016	0.4430	0.3746	0.4876
Vehicle	0.3713	0.4515	0.4539	0.4243	0.4090	0.3664	0.3522	0.3605
Image	0.5629	0.4595	0.6348	0.7181	0.7043	0.5190	0.5524	0.3505
Average	0.6105	0.6049	0.6227	**0.6288**	0.6141	0.5683	0.5334	0.5362

M is the number of attributes
Bold value indicates the best performance

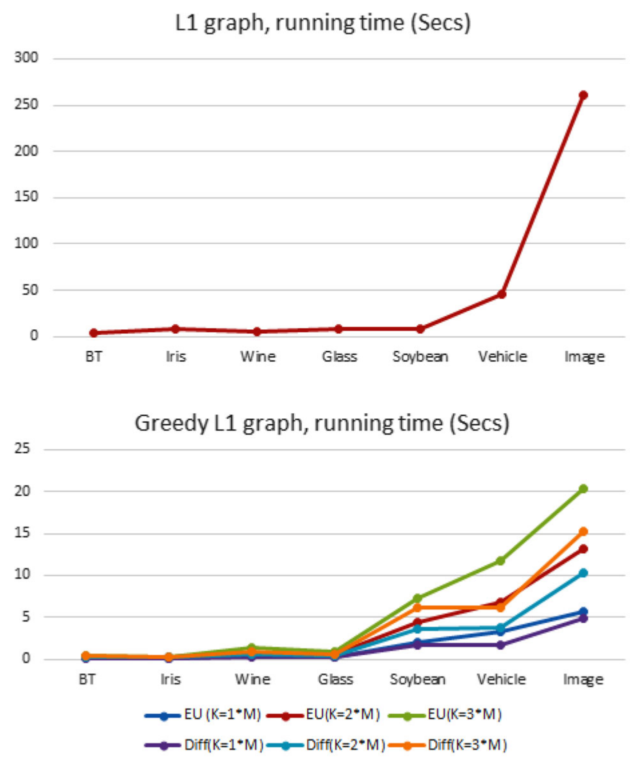

Fig. 7 Running time of different $\mathcal{L}_1$ graph construction algorithms. *Top* original $\mathcal{L}_1$ graph construction algorithm. *Bottom* the construction of $\mathcal{L}_1$ graph using greedy solver

the result of diffusion ranking is better than that of Euclidean distance ranking.

5.1.6 Running time

We report the running time of generating $\mathcal{L}_1$ graphs using different construction algorithms. As we can see from Fig. 7, the greedy-$\mathcal{L}_1$ graphs have consumed significantly less construction time than that in original $\mathcal{L}_1$ graphs.

5.1.7 Graph sparsity

We check the sparsity of graphs by calculating the edge density:

$$\text{Sparsity}(G) = \frac{|E|}{|V| * (|V| - 1)}. \qquad (10)$$

The results are reported in Table 5. We can see that greedy-$\mathcal{L}_1$ graphs with diffusion distance ranking are more sparse than that with Euclidean distance ranking.

5.2 Large size data and multiple classes data

In this section, we present the experimental results of three large datasets. To keep the integrity of our experiments, two multiple classes data are also examined.

5.2.1 Datasets

We select following datasets for our experiments. Three large size datasets are: first $2k$ testing images of MNIST (MNIST4K), COIL 100 objects database (COIL100) and USPS handwritten digit database (USPS). Two multiple classes datasets are: isolet spoken letter recognition dataset (ISOLET), extended Yale face database B (YaleB). The statistics of selected datasets can be described in Table 6.

5.2.2 Spectral clustering performance

The spectral clustering performance is shown in Tables 7 and 8. As we can see, Gaussian graphs have overall better performance than different $\mathcal{L}_1$ graphs. For the performance between original $\mathcal{L}_1$ graph (with OMP greedy solver) and greedy $\mathcal{L}_1$ graphs, the greedy version is better.

Table 5 Graph sparsity comparison of different graph construction algorithms

Name	$\mathcal{L}_1$ graph	Gaussian graph	Greedy-$\mathcal{L}_1$ graph (Euclidean)			Greedy-$\mathcal{L}_1$ graph (diffusion)		
			$K = 1 \times M$	$K = 2 \times M$	$K = 3 \times M$	$K = 1 \times M$	$K = 2 \times M$	$K = 3 \times M$
BT	0.0604	1	0.0457	0.0615	0.0705	0.0341	0.0442	0.0548
Iris	0.0403	1	0.0217	0.0288	0.0311	0.0203	0.0237	0.0265
Wine	0.0600	1	0.0413	0.0496	0.0552	0.0347	0.0409	0.0437
Glass	0.0369	1	0.0242	0.0308	0.0349	0.0188	0.0204	0.0239
Soybean	0.030	1	0.0286	0.0317	0.0346	0.0258	0.0299	0.034
Vehicle	0.0135	1	0.0104	0.0124	0.0135	0.0062	0.0074	0.0084
Image	0.0039	1	0.0034	0.004	0.0044	0.0026	0.0029	0.0027

M is the number of attributes

Table 6 The statistics of three large datasets and two multiple classes datasets

Name	#samples	#attributes	#clusters
ISOLET	1560	617	25
YaleB	2414	1024	38
MNIST4K	4000	784	9
COIL100	7200	1024	100
USPS	9298	256	10

5.2.3 Graph sparsity

We also check the sparsity of different similarity graphs. The result in Table 9 shows that greedy $\mathcal{L}_1$ graphs with diffusion ranking are more denser than other $\mathcal{L}_1$ graphs. And the ordinary $\mathcal{L}_1$ graph (OMP) has the lowest sparsity.

It is known that the sparsity of graph will affect the performance of graph cut and then to spectral clustering. And the spectral clustering performance will drop if the sparsity is lower than a threshold [17]. Since $\mathcal{L}_1$ graph is a sparse graph in nature, we want to know the correlation between the sparsity and clustering performance. To evaluate this, we choose the "USPS" dataset and generating graphs with different sparsity by setting the reconstruction approximation error bound to different thresholds. They are: [0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001, 0.00005, 0.00001]. For the size of "ranked dictionary," we choose size to $2M$ which is 512. The trend of spectral clustering performance with different sparsity can be shown as the left subplot of Fig. 8. We can see that when the sparsity value is lower than 0.0072 , the spectral clustering performance drops catastrophically. The relationship between the approximation error and the graph sparsity is presented at the right side of Fig. 8. By reading from the curve, we know that the approximation error and sparsity have a negative relationship. To maintain the greedy $\mathcal{L}_1$ as dense as possible, we need to set a lower bound of approximation error.

5.2.4 Running time

We also record the running time of building different similarity graphs. From Table 10, we see that the running time increases, while the data size becomes larger. However, the "USPS" has lesser running time than "COIL100" even its data size is bigger. The reason is that "USPS" has smaller number of features than "COIL100" and this causes the $\mathcal{L}_1$ solver to need more computation time for finding sparse solutions.

Table 7 NMI results of spectral clustering with different similarity graphs

Name	$\mathcal{L}_1$ (OMP)	Gaussian	Greedy-$\mathcal{L}_1$ graph (Euclidean)			Greedy-$\mathcal{L}_1$ graph (diffusion)		
			$K = 1 \times M$	$K = 2 \times M$	$K = 3 \times M$	$K = 1 \times M$	$K = 2 \times M$	$K = 3 \times M$
ISOLET	0.2571	0.7821	0.5501	0.4202	NA	0.1903	0.2993	NA
YaleB	0.2349	0.4219	0.2493	0.2895	NA	0.2003	0.4408	NA
MNIST4K	0.2503	0.4426	0.2679	0.1850	0.2438	0.0737	0.0333	0.0575
COIL100	0.3556	0.7726	0.7072	0.6533	0.6283	0.4044	0.4166	0.4788
USPS	0.1585	0.6580	0.6608	0.6571	0.6488	0.0360	0.0621	0.0399
Average	0.2513	**0.5457**	0.4713	0.4462	0.5070	0.1809	0.2504	0.1921

M is the number of attributes
Bold value indicates the best performance

Table 8 ACC results of spectral clustering with different similarity graphs

NAME	$\mathcal{L}_1$ (OMP)	Gaussian	Greedy-$\mathcal{L}_1$ graph (Euclidean)			Greedy-$\mathcal{L}_1$ graph (diffusion)		
			$K = 1 \times M$	$K = 2 \times M$	$K = 3 \times M$	$K = 1 \times M$	$K = 2 \times M$	$K = 3 \times M$
ISOLET	0.2038	0.6974	0.4205	0.3327	NA	0.1705	0.2558	NA
YaleB	0.1533	0.2618	0.2067	0.2606	NA	0.1831	0.4321	NA
MNIST4K	0.2787	0.5302	0.3900	0.2755	0.3538	0.1847	0.1685	0.1845
COIL100	0.1192	0.5201	0.4746	0.4368	0.4012	0.2381	0.2326	0.2778
USPS	0.2122	0.7018	0.6723	0.6740	0.6950	0.1590	0.1778	0.1663
Average	0.1934	**0.5423**	0.4328	0.3959	0.4833	0.1871	0.2534	0.2095

M is the number of attributes
Bold value indicates the best performance

Table 9 Graph sparsity results of different similarity graphs

Name	$\mathcal{L}_1$ (OMP)	Gaussian	Greedy-$\mathcal{L}_1$ graph (Euclidean)			Greedy-$\mathcal{L}_1$ graph (diffusion)		
			$K = 1 \times M$	$K = 2 \times M$	$K = 3 \times M$	$K = 1 \times M$	$K = 2 \times M$	$K = 3 \times M$
ISOLET	0.0010	1	0.3304	0.2679	NA	0.4288	0.2804	NA
YaleB	0.0019	1	0.1968	0.1713	NA	0.3797	0.1952	NA
MNIST4K	0.0043	1	0.1022	0.0954	0.0929	0.1470	0.1267	0.1076
COIL100	0.0002	1	0.0786	0.0620	0.0574	0.1887	0.1198	0.0929
USPS	0.0003	1	0.0076	0.0072	0.0071	0.0246	0.0225	0.0214

M is the number of attributes

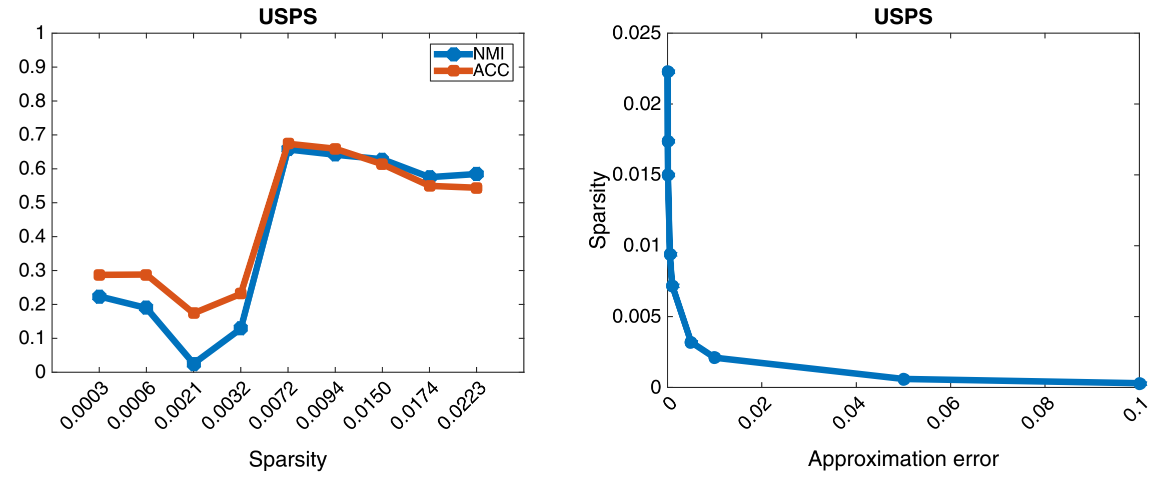

Fig. 8 The impact of graph sparsity to spectral clustering performance. *Left* graph sparsity versus NMI and ACC. *Right* $\mathcal{L}_1$ solver approximation error versus graph sparsity

Table 10 Running time of different similarity graphs

Name	$\mathcal{L}_1$ (OMP)	Gaussian	Greedy-$\mathcal{L}_1$ graph (Euclidean)			Greedy-$\mathcal{L}_1$ graph (diffusion)		
			$K = 1 \times M$	$K = 2 \times M$	$K = 3 \times M$	$K = 1 \times M$	$K = 2 \times M$	$K = 3 \times M$
ISOLET	243.9	1.1	202.5	310.6	NA	263.0	327.7	NA
YaleB	836.1	4.3	758.7	1187.6	NA	1097.9	1197.7	NA
MNIST4K	1435.8	9.8	814.8	1048.5	1341.9	848.9	1158.4	1412.7
COIL100	5541.3	36.1	2379.7	3225.0	5447.8	4108.5	5091.8	7475.3
USPS	2499.5	16.4	93.2	123.1	174.1	221.1	259.5	323.1

M is the number of attributes

6 Conclusion

In this paper, we have designed a greedy algorithm to construct $\mathcal{L}_1$ graph. Moreover, we introduced the concept of "ranked dictionary," which not only preserves the locality but also solves the curse of normalization. Moreover, it can construct $\mathcal{L}_1$ graph efficiently for large size data (#instances $\geq$3000.) Except for the Euclidean metric and diffusion metric that have been discussed in this paper, the user can choose other ranking methods such as manifold ranking that could be more appropriate for specific dataset in real applications. Our greedy algorithm can generate sparse $\mathcal{L}_1$ graph faster than the original $\mathcal{L}_1$ graph construction algorithm, and the resulting graphs have better clustering performance on average than original $\mathcal{L}_1$ graph. Nevertheless, our algorithm could be generalized in a straightforward way by introducing regularization terms such as elastic net into the current solver, which would indicate the quality of generated $\mathcal{L}_1$ graphs could be further improved.

Acknowledgments This research is supported in part by NSF (IIS-0949467, IIS-1047715 and IIS-1049448) and NSFC (61532002, 61190120, 61190125, 61190124). We thank the anonymous reviewers for their constructive critiques.

References

1. Brito, M., Chavez, E., Quiroz, A., Yukich, J.: Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection. Stat. Probab. Lett. **35**(1), 33–42 (1997)
2. Bruckstein, A.M., Elad, M., Zibulevsky, M.: On the uniqueness of nonnegative sparse solutions to underdetermined systems of equations. IEEE Trans. Inf. Theory **54**(11), 4813–4820 (2008)
3. Chen, S.S., Donoho, D.L., Saunders, M.A.: Atomic decomposition by basis pursuit. SIAM Rev. **43**(1), 129–159 (2001)
4. Cheng, B., Yang, J., Yan, S., Fu, Y., Huang, T.S.: Learning with-graph for image analysis. IEEE Trans. Image Process. **19**(4), 858–866 (2010)
5. Coifman, R.R., Lafon, S.: Diffusion maps. Appl. Comput. Harmonic Anal. **21**(1), 5–30 (2006)
6. Correa, C.D., Lindstrom, P.: Locally-scaled spectral clustering using empty region graphs. In: Proceedings of the 18th ACM SIGKDD international conference on Knowledge Discovery and Data mining, pp. 1330–1338. ACM (2012)
7. Donoser, M., Bischof, H.: Diffusion processes for retrieval revisited. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1320–1327. IEEE (2013)
8. Elhamifar, E., Vidal, R.: Sparse subspace clustering. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 2790–2797. IEEE (2009)
9. Elhamifar, E., Vidal, R.: Sparse subspace clustering: algorithm, theory, and applications. IEEE Trans. Pattern Anal. Mach. Intell. **35**(11), 2765–2781 (2013)
10. Fang, Y., Wang, R., Dai, B., Wu, X.: Graph-based learning via auto-grouped sparse regularization and kernelized extension. IEEE Trans. Knowl. Data Eng. **27**(1), 142–154 (2015)
11. Han, S., Huang, H., Qin, H., Yu, D.: Locality-preserving l1-graph and its application in clustering. In: Proceedings of the 30th Annual ACM Symposium on Applied Computing, pp. 813–818. ACM (2015)
12. Han, S., Qin, H.: A greedy algorithm to construct l1 graph with ranked dictionary. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 309–321. Springer (2016)
13. Kim, S.J., Koh, K., Lustig, M., Boyd, S., Gorinevsky, D.: An interior-point method for large-scale l 1-regularized least squares. IEEE J Sel. Top. Signal Process. **1**(4), 606–617 (2007)
14. Koh, K., Kim, S.J., Boyd, S.P.: An interior-point method for large-scale l1-regularized logistic regression. J. Mach. Learn. Res. **8**(8), 1519–1555 (2007)
15. Lin, T.H., Kung, H.: Stable and efficient representation learning with nonnegativity constraints. In: Proceedings of the 31st International Conference on Machine Learning (ICML-14), pp. 1323–1331 (2014)
16. Mairal, J., Bach, F., Ponce, J., Sapiro, G.: Online learning for matrix factorization and sparse coding. J. Mach. Learn. Res. **11**, 19–60 (2010)
17. Nadakuditi, R.R., Newman, M.E.: Graph spectra and the detectability of community structure in networks. Phys. Rev. Lett. **108**(18), 188,701 (2012)
18. Ng, A.Y., Jordan, M.I., Weiss, Y., et al.: On spectral clustering: analysis and an algorithm. Adv. Neural Inf. Process. Syst. **2**, 849–856 (2002)
19. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science **290**(5500), 2323–2326 (2000)
20. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **22**(8), 888–905 (2000)
21. Tibshirani, R.: Regression shrinkage and selection via the lasso. J. R. Stat. Soc. Ser. B (Methodol.) **58**(1), 267–288 (1996)
22. Tropp, J.A., Gilbert, A.C.: Signal recovery from random measurements via orthogonal matching pursuit. Inf. Theory, IEEE Trans. **53**(12), 4655–4666 (2007)
23. Tropp, J.A., Wright, S.J.: Computational methods for sparse solution of linear inverse problems. Proc. IEEE **98**(6), 948–958 (2010)
24. Von Luxburg, U.: A tutorial on spectral clustering. Stat. Comput. **17**(4), 395–416 (2007)
25. Wright, J., Yang, A.Y., Ganesh, A., Sastry, S.S., Ma, Y.: Robust face recognition via sparse representation. IEEE Trans. Pattern Anal. Mach. Intell. **31**(2), 210–227 (2009)
26. Yan, S., Wang, H.: Semi-supervised learning by sparse representation. In: Society for Industrial and Applied Mathematics. Proceedings of the SIAM International Conference on Data Mining, p. 792. Society for Industrial and Applied Mathematics (2009)
27. Yang, A.Y., Sastry, S.S., Ganesh, A., Ma, Y.: Fast l1-minimization algorithms and an application in robust face recognition: a review. In: 2010 17th IEEE International Conference on Image Processing (ICIP)., pp. 1849–1852. IEEE (2010)
28. Yang, Y., Feng, J., Yang, J., Huang, T.S.: Learning l1-graph for data clustering. CoRR (2015). arxiv:1510.08520
29. Yang, Y., Wang, Z., Yang, J., Han, J., Huang, T.: Regularized l1-graph for data clustering. In: Proceedings of the British Machine Vision Conference. BMVA Press (2014)
30. Yang, Y., Wang, Z., Yang, J., Wang, J., Chang, S., Huang, T.S.: Data clustering by laplacian regularized l1-graph. In: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, pp. 3148–3149 (2014)
31. You, C., Robinson, D., Vidal, R.: Scalable sparse subspace clustering by orthogonal matching pursuit. In: IEEE Conference on Computer Vision and Pattern Recognition, vol. 1 (2016)
32. Zhang, Y.M., Huang, K., Hou, X., Liu, C.L.: Learning locality preserving graph from data. IEEE Trans. Cybern. **44**(11), 2088–2098 (2014)
33. Zhou, G., Lu, Z., Peng, Y.: L1-graph construction using structured sparsity. Neurocomputing **120**, 441–452 (2013)