A Greedy Algorithm to Construct L1 Graph with Ranked Dictionary

Shuchu Han, Hong Qin

Stony Brook University (SUNY), USA
{shhan,qin}@cs.stonybrook.edu

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 metric, and another is based on diffusion metric.

Keywords: Sparse graph, clustering

1 Introduction

For graph-oriented learning tasks, a quality graph representation [4] 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-shape 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. [2]. Based on existing sparse representation frameworks [10] [12], the construction algorithm of $\mathcal{L}_1$ graph can be described as follows: Given an input data samples $\mathbf{X} = [\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_n}]$, where each $\boldsymbol{x_i}, i \in [1, \cdots, n]$ is a vector that represents one single data sample. The $\mathcal{L}_1$

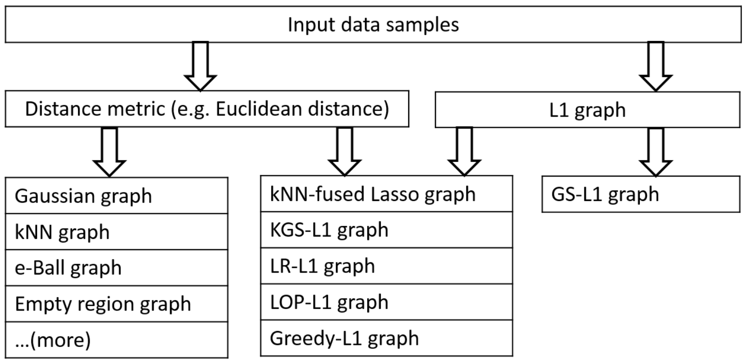

Fig. 1. Connection of Greedy $\mathcal{L}_1$ graph to other graphs. Several of them are: kNN-fused Lasso graph [16], Group Sparse (GS) $\mathcal{L}_1$ graph, Kernelized Group Sparse (KGS) $\mathcal{L}_1$ graph [6], Laplacian Regularized (LR) $\mathcal{L}_1$ graph [14] and Locality Preserving (LOP) $\mathcal{L}_1$ graph [7].

graph of $\mathbf{X}$ is built by finding a sparse coding [11] 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:

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

where dictionary $\boldsymbol{\Phi}^i = [\boldsymbol{x_1}, \cdots, \boldsymbol{x_{i-1}}, \boldsymbol{x_{i+1}}, \cdots, \boldsymbol{x_n}]$, 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. [2] show that $\mathcal{L}_1$ graph has three advantages comparing to 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, semi-supervised learning, etc [2]. 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 [2]; (4) lost of the locality [6] [15] [7]. 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*.

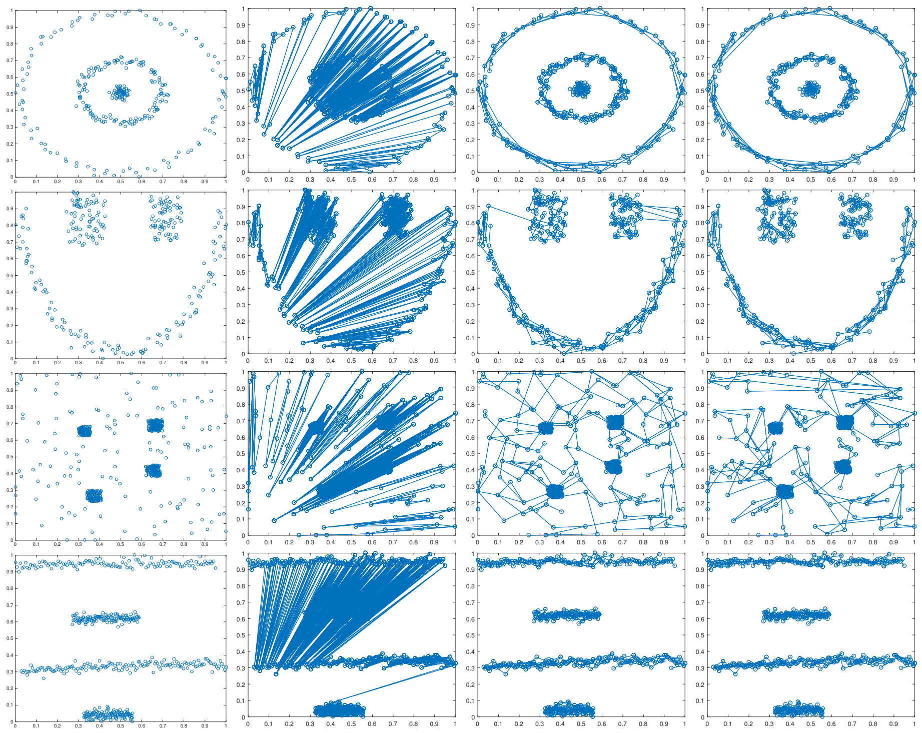

Fig. 2. $\mathcal{L}_1$ graphs generated by different construction algorithms. From left to right: 2D toy dataset; $\mathcal{L}_1$ graph; Greedy-$\mathcal{L}_1$ graph with Euclidean metric (K=15); Greedy-$\mathcal{L}_1$ graph with Diffusion metric (K=15).

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 [16] preserves the local structure information of input data, the auto-grouped sparse regularization [6] adds the group effect to the final graph, and the Graph Laplacian regularization [13] [14] 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 [7].

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, group effects, etc. 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 non-negative 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 [3]. Our algorithm has significant time-reduction about generating $\mathcal{L}_1$ graphs. Comparing to the original $\mathcal{L}_1$ graph construction method, our algorithm loses the nonparametric characteristics and is only offering a sub-optimal 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 Figure 1. The main contributions of our paper can be summarized by

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 sub-optimal solution.

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 Section 2. Second, we will introduce our proposed greedy algorithm in Section 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 Section 5 and draw conclusion in Section 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) non-local edges.

2.1 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 is not

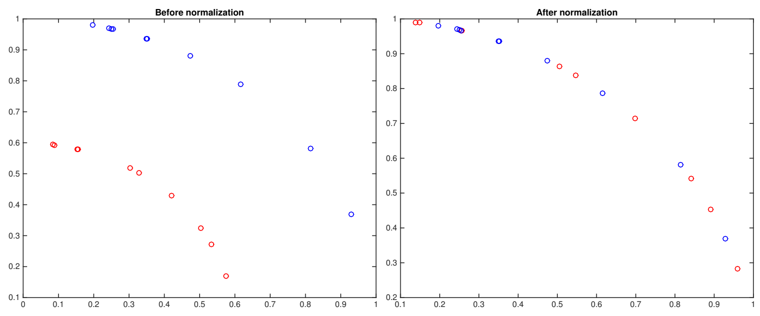

Fig. 3. Demonstration of dictionary normalization for a toy dataset. The red and blue points represent two different clusters. Left: before normalization; right: after normalization. We can see that the neighborhood relationship is changed after normalization.

preserving 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.

2.2 Non-local Edges

During the construction of $\mathcal{L}_1$ graph, an over-complete dictionary is required 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 non-local edges. In other words, it doesn't preserve the **locality** of input data [7]. This phenomenon can be illustrated in Fig. 4,

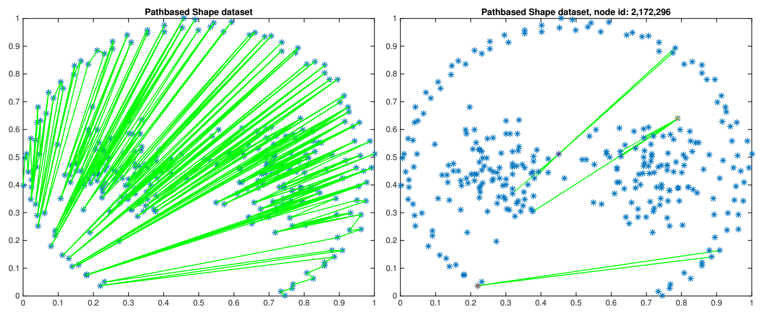

Fig. 4. $\mathcal{L}_1$-graph of path-based dataset. Left: the entire graph; right: edges of three selected points. We can see the existence of non-local edges.

3 New Algorithm

In this section, we introduce the concept of ranked dictionary with two different strategies: Euclidean metric and diffusion metric. Furthermore, we present our proposed greedy algorithm and describe how to generate Greedy-$\mathcal{L}_1$ graph from it.

3.1 Ranked Dictionary

The use of k-nearest neighbors as dictionary is proved to have better quality than original $\mathcal{L}_1$ graph [7]. However, it can not solve the dilemma that there might exist data samples with the same direction but different length in input data. The dictionary normalization process will project them onto to the same location at hypersphere. Since they have the same values, the $\mathcal{L}_1$ minimization solver will choose one of them randomly. To avoid this randomness, we need to rank those atoms (or data samples) of dictionary.

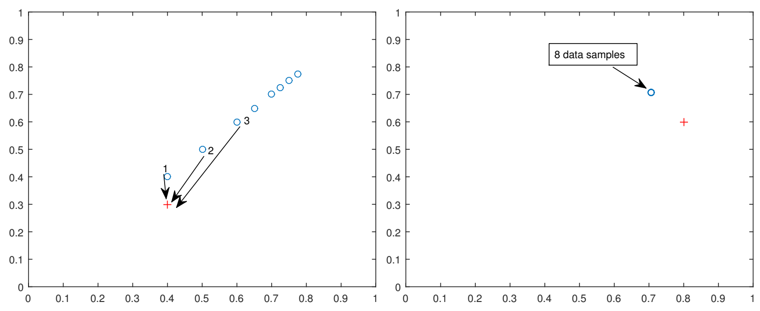

Fig. 5. Ranked dictionary. Left: eight data samples have the same direction but with different length. Red cross is the target data sample for calculating sparse coefficients. Right: after normalization, those eight data samples have the same location.

Euclidean Metric. 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:

$$dist(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 \tag{2}$$

Diffusion Metric. As pointed out by Yang et al. [14], 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 [3] [5], we rank the atoms of dictionary through diffusion matrix. A diffusion process has three stages [5]: (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:

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

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

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

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 is selected as dictionary. The algorithmic details can be documented as follows:

Algorithm 1: Diffusion Dictionary

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

3.2 Greedy-$\mathcal{L}_1$ Graph

We now propose a greedy algorithm to build $\mathcal{L}_1$ graph. Our proposed algorithm is based on non-negative orthogonal matching pursuit (NNOMP) [1] [9]. By using this solver, we switch the $\mathcal{L}_1$ minimization problem (**P1**) back to the original $\mathcal{L}_0$ optimization with non-negative constraints (**P2**) as:

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

The main difference between our algorithm and the original NNOMP [1] is that the atoms of dictionary are ranked. We let the solver choose and assign higher coefficient values to atoms that are closer to source data sample. The detailed processes are described in Algorithm 2.

Algorithm 2: Greedy Solver

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

4 Related Works

Original $\mathcal{L}_1$ graph [2] is a pure numerical result and doesn't 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 [6], OSCAR regularization [6], and Graph Laplacian regularization [14].

Another research direction of $\mathcal{L}_1$ graph is to reduce its high computational cost. Zhou et al. [16] 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. [6]

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.

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

Table 1. Data set statistics.

5 Experiments

We now present our experimental results. We first document our configuration of parameters and datasets. Second, we evaluate the effectiveness of our proposed graph construction methods through spectral clustering application. 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 PC with Intel 4-core 3.4GHz CPU and 16GB RAM.

5.1 Experimental Setup

Datasets. To demonstrate the performance of our proposed algorithm, we evaluate it on seven UCI benchmark datasets including three biological data sets (BreastTissue, Iris, Soybean), two vision image data sets (Vehicle, Image), one chemistry data set (Wine), and one physical data set (Glass), whose statistics are summarized in Table 1. All of these data sets have been popularly used in spectral clustering analysis research. The diverse combinations of data sets are necessary for our comprehensive studies.
Parameters Setting. In our experiments, we use the *l1_ls* solver [8] for original $\mathcal{L}_1$ graph construction algorithms. We set the solver's parameter λ to 0.1. The *threshold* $\theta_{threshold}$ of Greedy solver 2 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 of data attribute size.

5.2 Spectral Clustering Performance

Baseline. To evaluate the quality of our algorithms, we compare the spectral clustering performance with Gaussian similarity graph, and original $\mathcal{L}_1$ graph. The results are documented in Table 2 and Table 3.

Name	$\mathcal{L}_1$	Gaussian			Greedy-$\mathcal{L}_1$ graph (Euclidean)			Greedy-$\mathcal{L}_1$ graph (Diffusion)		
		K=1*M	K=2*M	K=3*M	K=1*M	K=2*M	K=3*M	K=1*M	K=2*M	K=3*M
BT	0.4582	0.3556	0.4909	0.4722	0.5473	0.4517	0.5024	0.4197	0.4073	0.3839
Iris	0.5943	0.4557	0.5923	0.7696	0.3950	0.4623	0.4070	0.5106	0.4626	0.4640
Wine	0.7717	0.8897	0.8897	0.8897	0.8943	0.9072	0.8566	0.6925	0.4291	0.6093
Glass	0.3581	0.1598	0.2941	0.2614	0.2569	0.3688	0.3039	0.2991	0.3056	0.2918
Soybean	0.7373	0.6839	0.6911	0.6541	0.6919	0.6833	0.6775	0.5788	0.5493	0.5432
Vehicle	0.1044	0.1528	0.1519	0.1341	0.1512	0.2121	0.2067	0.1438	0.1035	0.1244
Image	0.4969	0.2461	0.3382	0.0486	0.5821	0.6673	0.6649	0.4866	0.4483	0.3155
Average	0.5030	0.4205	0.4926	0.4614	0.5170	**0.5361**	0.5170	0.4473	0.3865	0.3903

Table 2. NMI comparison of graph construction algorithms. M is the number of attributes.

Name	$\mathcal{L}_1$	Gaussian			Greedy-$\mathcal{L}_1$ graph (Euclidean)			Greedy-$\mathcal{L}_1$ graph (Diffusion)		
		K=1*M	K=2*M	K=3*M	K=1*M	K=2*M	K=3*M	K=1*M	K=2*M	K=3*M
BT	0.5472	0.3208	0.5189	0.5472	0.6698	0.4811	0.5943	0.4528	0.4906	0.4717
Iris	0.7400	0.6667	0.6867	0.9090	0.6933	0.7200	0.6800	0.7200	0.6533	0.64
Wine	0.9326	0.9719	0.9719	0.9719	0.9719	0.9719	0.9551	0.8989	0.7865	0.8596
Glass	0.4206	0.4206	0.4486	0.4206	0.4579	0.4533	0.4346	0.4626	0.4813	0.5187
Soybean	0.6156	0.5440	0.5570	0.5505	0.5244	0.4853	0.5016	0.4430	0.3746	0.4876
Vehicle	0.3713	0.3983	0.3983	0.4066	0.4539	0.4243	0.4090	0.3664	0.3522	0.3605
Image	0.5629	0.3262	0.3919	0.1895	0.6348	0.7181	0.7043	0.5190	0.5524	0.3505
Average	0.6105	0.5546	0.5757	0.5746	0.6227	**0.6288**	0.6141	0.5683	0.5334	0.5362

Table 3. AC comparison of different graph construction algorithms. M is the number of attributes.

Evaluation Metrics. We evaluate the spectral clustering performance with Normalized Mutual Information (NMI) and Accuracy (AC). 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 are correct. It's value also ranges from 0 to 1, and the higher the better.

Greedy-$\mathcal{L}_1$ Graph vs. Gaussian Graph. Overall, the Greedy-$\mathcal{L}_1$ graph using Euclidean metric has better average spectral clustering performance than Gaussian graphs. However, since the Gaussian graph we used are not tuned, the best clustering performance of Gaussian graphs may not occur in our experiments.

Greedy-$\mathcal{L}_1$ Graph vs. $\mathcal{L}_1$ 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, AC=0.74); Soybean (NMI=0.7373, AC=0.6156). The best result of Greedy-$\mathcal{L}_1$ graph are: Iris (NMI=0.5106, AC=0.72); Soybean (NMI=0.6919, AC=0.5244).

Euclidean Metric vs. Diffusion Metric. The Euclidean metric appears to have better clustering performance than that of diffusion metric in general. This is rather a surprising result to us. Only for Iris dataset, the result of diffusion metric is better than that of Euclidean metric.

5.3 Discussions

Running Time. We report the running time of generating $\mathcal{L}_1$ graphs using different construction algorithms. As we can see from Fig. 6, the Greedy-$\mathcal{L}_1$

graphs have consumed significantly less construction time than that in original $\mathcal{L}_1$ graphs.

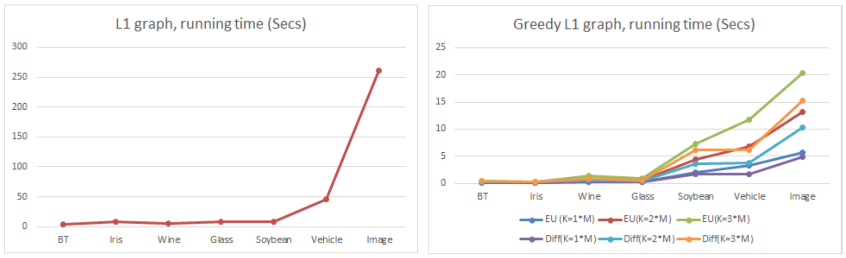

Fig. 6. Running time of different $\mathcal{L}_1$ graph construction algorithms. Left: original $\mathcal{L}_1$ graph construction algorithm. Right: the construction of $\mathcal{L}_1$ graph using greedy solver.

Graph Sparsity. We check the sparsity of graphs by calculating the edge density:

$$sparsity(G) = \frac{|E|}{|V| * (|V| - 1)}. \tag{8}$$

The results are reported in Table 4. We can see that Greedy-$\mathcal{L}_1$ graphs with diffusion metric are more sparse than that with Euclidean metric.

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

Table 4. Graph sparsity comparison of different graph construction algorithms. M is the number of attributes.

6 Conclusion

In this paper, we have devised a greedy algorithm to construct $\mathcal{L}_1$ graph. Moreover, we introduced the concept of ranked dictionary for our greedy solver. 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. Bruckstein, A.M., Elad, M., Zibulevsky, M.: On the uniqueness of nonnegative sparse solutions to underdetermined systems of equations. IEEE Transactions on Information Theory, 54(11), 4813–4820 (2008)
2. Cheng, B., Yang, J., Yan, S., Fu, Y., Huang, T.S.: Learning with-graph for image analysis. IEEE Transactions on Image Processing, 19(4), 858–866 (2010)
3. Coifman, R.R., Lafon, S.: Diffusion maps. Applied and Computational Harmonic Analysis 21(1), 5–30 (2006)
4. 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)
5. Donoser, M., Bischof, H.: Diffusion processes for retrieval revisited. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 1320–1327. IEEE (2013)
6. Fang, Y., Wang, R., Dai, B., Wu, X.: Graph-based learning via auto-grouped sparse regularization and kernelized extension. IEEE Transactions on Knowledge and Data Engineering, 27(1), 142–154 (2015)
7. 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)
8. Koh, K., Kim, S.J., Boyd, S.P.: An interior-point method for large-scale l1-regularized logistic regression. Journal of Machine Learning Research 8(8), 1519–1555 (2007)
9. 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)
10. Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological) pp. 267–288 (1996)
11. Tropp, J.A., Wright, S.J.: Computational methods for sparse solution of linear inverse problems. Proceedings of the IEEE 98(6), 948–958 (2010)
12. Wright, J., Yang, A.Y., Ganesh, A., Sastry, S.S., Ma, Y.: Robust face recognition via sparse representation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(2), 210–227 (2009)
13. 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)
14. 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)
15. Zhang, Y.M., Huang, K., Hou, X., Liu, C.L.: Learning locality preserving graph from data. IEEE Transactions on Cybernetics 44(11), 2088–2098 (2014)
16. Zhou, G., Lu, Z., Peng, Y.: L1-graph construction using structured sparsity. Neurocomputing 120, 441–452 (2013)