Clustering with Self-Learned Graph Regression

Lai Wei*, **Jin Liu**

College of Information Engineering, Shanghai Maritime University
Haigang Avenue 1550, Shanghai, China
weilai@shmtu.edu.cn, liujin@shmtu.edu.cn

Abstract

Graph-based clustering algorithms aim to construct an affinity graph that accurately captures the intrinsic structure of a dataset. To achieve this goal, these algorithms often use the k-nearest-neighbor (k-nn) method to build a graph regularizer for the required affinity graph, enabling it to have a grouping effect. However, due to the complex nature of real-world data, the k-nn method often fails to capture the true neighborhood relationships of a dataset, which in turn limits the quality of the learned affinity graph. Motivated by the insight that a learned affinity graph itself can more effectively reflect the underlying data structure, we propose a new graph-based clustering method, termed Self-learned Graph Regression (SGR). Unlike traditional approaches, SGR constructs its graph regularizer directly from the affinity graph being learned, allowing the graph to adaptively capture more accurate structural information. To solve the proposed problem, we develop an optimization algorithm along with an acceleration strategy. We further analyze the convergence and computational complexity of the proposed algorithm. Extensive clustering experiments on various benchmark datasets demonstrate that our method outperforms the state-of-the-art graph-based clustering algorithms.

Code — https://github.com/weilyshmtu/SGR

Introduction

Clustering is a fundamental task in machine learning that aims to partition data samples into distinct groups such that samples within the same group exhibit high similarity, while those in different groups exhibit low similarity. In recent years, clustering techniques have been widely applied across various domains, including image processing (Lei et al. 2019), object reidentification (Zhang et al. 2021), and social network analysis (Lee et al. 2019). Consequently, a wide range of clustering algorithms have been developed, such as k-means related clustering (Pei et al. 2023), hierarchical clustering (Sarfraz, Sharma, and Stiefelhagen 2019), density-based clustering (Fu et al. 2024), and graph-based clustering (Wang et al. 2025; Wei et al. 2025). Among these methods, graph-based clustering has demonstrated superior performance and has attracted considerable attention.

*Corresponding author

Graph-based clustering aims to construct an affinity graph that captures the underlying cluster structure within a dataset. Once the affinity graph is constructed, a spectral clustering algorithm—such as Normalized Cuts (Ncuts) (Shi and Malik 2000)—is employed to generate the final clustering results. Let $\mathbf{X} \in \mathbb{R}^{n\times d}$ denote the data matrix, where n is the number of samples and d is the feature dimension. A general formulation of graph-based clustering can be written as:

$$\min_{\mathbf{S}} \mathcal{L}(\mathbf{X}, \mathbf{S}) + \alpha\mathcal{R}(\mathbf{S}), \tag{1}$$

where $\mathbf{S} \in \mathbb{R}^{n\times n}$ represents the learned affinity matrix. $\mathcal{L}(\mathbf{X}, \mathbf{S})$ is a graph-learning term, which encourages the affinity graph to learn cluster structure information of the dataset, while $\mathcal{R}(\mathbf{S})$ denotes the graph regularization, which imposes desirable constraints on the learned graph. The parameter $\alpha > 0$ serves as a trade-off to balance these two components.

Although recent methods have incorporated feature learning strategies, such as dimensionality reduction (Patel, Nguyen, and Vidal 2013), graph filtering (Wei et al. 2023), and deep learning approaches (Ji et al. 2017), to extract latent representations from original data for improved clustering performance, the core of advanced graph-based clustering algorithms still lies in the design of $\mathcal{L}(\mathbf{X}, \mathbf{S})$ and $\mathcal{R}(\mathbf{S})$. Consequently, how to effectively construct these two terms remains a central focus of ongoing research.

According to the existing literature, $\mathcal{L}(\mathbf{X}, \mathbf{S})$ is commonly formulated in one of two ways: (1) as a self-representation term, $\|\mathbf{X} - \mathbf{S}\mathbf{X}\|_l$, where $\|\cdot\|_l$ denotes a certain matrix norm; or (2) as an adaptive neighborhood assignment term, $\sum_{i,j} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \mathbf{S}_{ij}$, where $\mathbf{x}_i$ and $\mathbf{x}_j$ are the i-th and j-th data samples in $\mathbf{X}$, and $\mathbf{S}_{ij}$ represents the entry (i, j)-th of $\mathbf{S}$.

The regularization term $\mathcal{R}(\mathbf{S})$ usually plays a crucial role in the construction of a high-quality affinity graph[1]. Accordingly, a variety of regularization strategies have been proposed from different perspectives to ensure that the learned graph possesses desirable properties, such as sparsity (Elhamifar and Vidal 2013; Liu et al. 2023), low-rankness (Liu et al. 2013; Ding et al. 2025), block-diagonal structure (Lu

[1]Here, the quality of a graph refers to its ability to accurately capture the underlying cluster structure of the dataset.

et al. 2019; Taştan, Muma, and Zoubir 2024), and locality-preserving property (also known as grouping effect) (Yin, Gao, and Lin 2016; Chen et al. 2022).

Among them, methods incorporating locality-preserving constraints have demonstrated promising experimental results. These methods are designed to ensure that the learned affinity graph maintains the grouping effect of the original data. That is, for data points that are close in the original feature space, their corresponding rows (or columns) in the affinity matrix are also similar.

A typical strategy for enforcing this property is through the application of a graph regression operator. Specifically, a k-nn method is employed to construct an adjacency matrix $\mathbf{W} \in \mathbb{R}^{n\times n}$ over the input data, from which the graph Laplacian matrix is derived as $\mathbf{L_W} = \mathbf{D_W} - (\mathbf{W} + \mathbf{W}^\top)/2$, where $\mathbf{W}^\top$ denotes the transpose of $\mathbf{W}$ and $\mathbf{D_W}$ is a diagonal degree matrix with $\mathbf{D_W}_{ii} = \sum_j (\mathbf{W}_{ij} + \mathbf{W}_{ji})/2$ for $1 \le i, j \le n$. Subsequently, the graph regularizer of $\mathbf{S}$ is defined as $\mathcal{R}(\mathbf{S}) = \mathrm{Tr}(\mathbf{S}^\top \mathbf{L_W} \mathbf{S})$, where $\mathrm{Tr}(\cdot)$ represents the matrix trace operator.

However, the locality-preserving constraints constructed by k-nn graph may degrade the quality of the final learned affinity matrix. Because, in practice, real-world datasets often exhibit complex and heterogeneous structures that cannot be adequately captured by a predefined k-nn graph. Intuitively, if such a predefined k-nn graph could be sufficient to capture the true data structure, there would be no need to learn a refined graph. A more effective strategy is to construct the locality-preserving constraints directly from the evolving affinity graph itself. This enables the learned graph to better reflect the true neighborhood structure, thereby enhancing the overall clustering performance.

Motivated by these observations, in this paper, we propose a novel graph-based clustering method, termed Self-learned Graph Regression (SGR). The key contributions of this work are as follows:

- We introduce a new self-learned graph regularizer within SGR, in which the affinity graph being learned is used to define its own locality-preserving constraint.
- We formulate SGR as a convex optimization problem w.r.t. each required variable and propose an acceleration strategy to enhance computational efficiency.
- We conduct extensive experiments on benchmark datasets, demonstrating that SGR consistently outperforms existing state-of-the-art graph-based clustering methods.

Related Works

Compared with self-representation-based algorithms, methods employing adaptive neighbor assignment are generally more computationally efficient. Therefore, we begin by reviewing the classical CAN method (Clustering with Adaptive Neighbors) proposed in (Nie, Wang, and Huang 2014). Using consistent notation, the CAN model can be formulated as follows:

$$\begin{aligned} \min_{\mathbf{S}} \quad & \textstyle\sum_{i,j} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \mathbf{S}_{ij} + \alpha \|\mathbf{S}\|_F^2 \\ s.t. \quad & \mathbf{S}\mathbf{1}_n = \mathbf{1}_n, \mathbf{S}_{ij} \ge 0, rank(\mathbf{L_S}) = n - c, \end{aligned} \tag{2}$$

where $\mathbf{1}_n = [1, \cdots, 1]^\top \in \mathbb{R}^{n\times 1}$, and $\|\mathbf{S}\|_F^2$ denotes the Frobenius norm of $\mathbf{S}$. The matrix $\mathbf{L_S}$ is the graph Laplacian of $\mathbf{S}$, namely $\mathbf{L_S} = \mathbf{D_S} - (\mathbf{S} + \mathbf{S}^\top)/2$ and $\mathbf{D_S}_{ii} = \sum_j (\mathbf{S}_{ij} + \mathbf{S}_{ji})/2$. The constraint $rank(\mathbf{L_S}) = n - c$ ensures that the affinity matrix $\mathbf{S}$ induces exactly c connected components, where c is the number of clusters in the dataset.

To address the rank constraint during optimization, the objective function of CAN is relaxed to be:

$$\min_{\mathbf{S}} \quad \textstyle\sum_{i,j} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \mathbf{S}_{ij} + \alpha \|\mathbf{S}\|_F^2 + \lambda \mathrm{Tr}(\mathbf{F}^\top \mathbf{L_S} \mathbf{F}), \tag{3}$$

where $Tr(\mathbf{F}^\top \mathbf{L_S} \mathbf{F}) = \sum_{i=1}^{c} \sigma_i(\mathbf{L_S})$, $\sigma_i(\mathbf{L_S})$ denotes the i-th smallest eigenvalue of $\mathbf{L_S}$, λ is adaptively tuned in the optimized algorithm[2]. And $\mathbf{F} \in \mathbb{R}^{n\times c}$, $\mathbf{F}^\top \mathbf{F} = \mathbf{I}_c$ (with $\mathbf{I}_c \in \mathbb{R}^{c\times c}$ being the identity matrix).

To accelerate convergence, CAN initializes $\mathbf{S}$ as follows:

$$\mathbf{S}_{ij} = \begin{cases} \frac{d_{ik+1} - d_{ij}}{k d_{ik+1} - \sum_{l=1}^{k} d_{il}}, & \text{if } j \le k; \\ 0, & \text{if } j > k; \end{cases} \tag{4}$$

where k is the number of nearest-neighbor samples and $d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2^2$.

Owing to its simplicity and effectiveness, several extensions of CAN have been proposed in recent years. For instance, Wang et al. (Wang et al. 2022) introduced entropy regularization as a replacement for the Frobenius norm regularizer in the original CAN framework. Wu et al. (Wu et al. 2023) proposed using similarity measures instead of Euclidean distances to construct the graph learning function, thus improving adaptability to complex data distributions. Nie et al. (Nie et al. 2020) proposed a self-weighted version of CAN that assigns different weights to features when computing pairwise Euclidean distances of samples. Additionally, Wang et al. (Wang et al. 2025) incorporated a doubly stochastic constraint on $\mathbf{S}$ produced by CAN, which further improves its performance.

Methodology

Motivation

Based on our investigation of CAN and its extensions, we identify two primary limitations in these algorithms: 1) They do not explicitly encourage the affinity matrix $\mathbf{S}$ to exhibit a grouping effect; 2) The hard-rank constraint imposed on $\mathbf{S}$ may reduce its flexibility and hinder its ability to accurately capture the underlying structure of the data (Lu et al. 2019).

To enhance the grouping effect of $\mathbf{S}$, a straightforward improvement is to use a graph-based regularizer of the form $\mathcal{R}(\mathbf{S}) = \mathrm{Tr}(\mathbf{S}^\top \mathbf{L_W} \mathbf{S})$. However, as discussed in the previous section, graph regularizers based on a predefined k-nn graph $\mathbf{W}$ can adversely affect the learning process of the affinity matrix $\mathbf{S}$.

Suppose that an ideal affinity matrix $\mathbf{S}$ has already been learned. It would naturally provide a more faithful representation of the local neighborhood relationships within the data compared to a predefined k-nn graph. Motivated by this insight, a self-learned graph regularizer can be defined as

[2]Usually, if the connected components of $\mathbf{S}$ is greater than c in current iteration, $\lambda \leftarrow \frac{\lambda}{2}$, and if it is less than c, $\lambda \leftarrow 2\lambda$.

$\mathcal{R}(\mathbf{S}) = \mathrm{Tr}(\mathbf{S}^\top \mathbf{L_S} \mathbf{S})$. This leads to the following optimization problem:

$$\begin{array}{ll} \min_{\mathbf{S}} & \sum_{i,j} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \mathbf{S}_{ij} + \alpha \mathrm{Tr}(\mathbf{S}^\top \mathbf{L_S} \mathbf{S}) \\ s.t. & \mathbf{S}\mathbf{1}_n = \mathbf{1}_n,\ \mathbf{S}_{ij} \geq 0, \mathbf{S}_{ii} = 0,\ \mathbf{S} = \mathbf{S}^\top, \\ & rank(\mathbf{L_S}) = n - c. \end{array} \tag{5}$$

Compared with CAN (i.e., Problem (2)), this formulation introduces three modifications:

- The Frobenius norm regularizer is replaced with a self-learned graph regularizer, which adapts to the evolving affinity structure;
- A constraint $\mathbf{S}_{ii} = 0$ is imposed to prevent trivial solutions such as $\mathbf{S} = \mathbf{I}_n$, where $\mathbf{I}_n$ is the identity matrix;
- More importantly, symmetry is enforced on $\mathbf{S}$, promoting its doubly stochastic property. Recent studies have shown that doubly stochastic affinity matrices significantly improve the performance of Ncuts (Wei et al. 2024; Wang et al. 2025). Furthermore, in our proposed formulation, the doubly stochastic nature of $\mathbf{S}$ also facilitates the optimization process, contributing to both algorithmic efficiency and clustering performance.

Given the doubly stochastic nature of $\mathbf{S}$, it follows that $\mathbf{L_S} = \mathbf{I}_n - \mathbf{S}$. Hence, the following equalities hold:

$$\begin{aligned} \textstyle\sum_{i,j} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \mathbf{S}_{ij} &= 2\mathrm{Tr}\big(\mathbf{X}^\top \mathbf{L_S} \mathbf{X}\big) \\ &= 2\mathrm{Tr}(\mathbf{X}^\top \mathbf{X}) - 2\mathrm{Tr}(\mathbf{X}^\top \mathbf{S} \mathbf{X}), \\ \mathrm{Tr}(\mathbf{S}^\top \mathbf{L_S} \mathbf{S}) &= \mathrm{Tr}(\mathbf{S}^\top \mathbf{S}) - \mathrm{Tr}(\mathbf{S}^\top \mathbf{S} \mathbf{S}) \\ &= \mathrm{Tr}(\mathbf{S}^2) - \mathrm{Tr}(\mathbf{S}^3). \end{aligned} \tag{6}$$

Then it can be seen that Problem (5) is non-convex with respect to $\mathbf{S}$, which poses challenges for efficient optimization. To overcome this difficulty, we introduce a variable $\mathbf{C} \approx \mathbf{S}$ and reformulate the problem into the following relaxed **Self-learned Graph Regression (SGR)** model:

$$\begin{array}{ll} \min_{\mathbf{S},\mathbf{C}} & \mathrm{Tr}(\mathbf{X}^\top \mathbf{L_S} \mathbf{X}) + \alpha \mathrm{Tr}(\mathbf{S}^\top \mathbf{L_C} \mathbf{S}) + \beta \|\mathbf{S} - \mathbf{C}\|_F^2 \\ s.t. & \mathbf{S}\mathbf{1}_n = \mathbf{1}_n,\ \mathbf{S}_{ij} \geq 0,\ \mathbf{S}_{ii} = 0, \mathbf{S} = \mathbf{S}^\top, \\ & \mathbf{C}\mathbf{1}_n = \mathbf{1}_n,\ \mathbf{C}_{ij} \geq 0,\ \mathbf{C}_{ii} = 0, \mathbf{C} = \mathbf{C}^\top, \\ & rank(\mathbf{L_C}) = n - c. \end{array} \tag{7}$$

where $\mathbf{L_C}$ is a Laplacian matrix constructed by $\mathbf{C}$. We also let $\mathbf{C}$ satisfy the doubly stochastic constraint, then $\mathbf{L_C} = \mathbf{I}_n - \mathbf{C}$. $\beta \geq 0$ is a hyper-parameter. Importantly, when either $\mathbf{S}$ or $\mathbf{C}$ is fixed, Problem (7) becomes convex with respect to the other variable, allowing efficient alternating minimization.

It is also important to note that, unlike existing CAN-based algorithms, we impose the hard-rank constraint on $\mathbf{C}$ rather than on $\mathbf{S}$, thereby allowing $\mathbf{S}$ to retain better flexibility. Furthermore, from the perspective of self-constraint clustering (Bai, Qi, and Liang 2023; Bai, Liang, and Zhao 2023), $\mathbf{C}$ can be interpreted as a learned structural constraint that guides $\mathbf{S}$ to get an accurate grouping effect.

Optimization

Now we will show how to solve the proposed SGR problem (i.e., Problem (7)). First, we apply a similar strategy in CAN to handle the constraint $rank(\mathbf{L_C}) = n - c$. Namely, with a large enough λ, Problem (7) is transferred to into the following problem:

$$\begin{array}{ll} \min_{\mathbf{S},\mathbf{C},\mathbf{F}} & \mathrm{Tr}(\mathbf{X}^\top \mathbf{L_S} \mathbf{X}) + \alpha \mathrm{Tr}(\mathbf{S}^\top \mathbf{L_C} \mathbf{S}) + \beta \|\mathbf{S} - \mathbf{C}\|_F^2 \\ & + \lambda \mathrm{Tr}(\mathbf{F}^\top \mathbf{L_C} \mathbf{F}) \\ s.t. & \mathbf{S}\mathbf{1}_n = \mathbf{1}_n,\ \mathbf{S}_{ij} \geq 0,\ \mathbf{S}_{ii} = 0, \mathbf{S} = \mathbf{S}^\top, \\ & \mathbf{C}\mathbf{1}_n = \mathbf{1}_n,\ \mathbf{C}_{ij} \geq 0, \mathbf{C}_{ii} = 0, \mathbf{C} = \mathbf{C}^\top, \\ & \mathbf{F}^\top \mathbf{F} = \mathbf{I}_c. \end{array} \tag{8}$$

Then we alternatively update the variables, $\mathbf{S}$, $\mathbf{C}$ and $\mathbf{F}$.

1. S-subproblem Assuming that $\mathbf{C}$ and $\mathbf{F}$ are given, Problem (8) can be simplified as follows:

$$\begin{array}{ll} \min_{\mathbf{S}} & \mathrm{Tr}(\mathbf{X}^\top \mathbf{L_S} \mathbf{X}) + \alpha \mathrm{Tr}(\mathbf{S}^\top \mathbf{L_C} \mathbf{S}) + \beta \|\mathbf{S} - \mathbf{C}\|_F^2 \\ \Leftrightarrow \min_{\mathbf{S}} & -\mathrm{Tr}(\mathbf{X}^\top \mathbf{S} \mathbf{X}) + \alpha \mathrm{Tr}(\mathbf{S}^\top \mathbf{L_C} \mathbf{S}) + \beta \|\mathbf{S} - \mathbf{C}\|_F^2 \\ \Leftrightarrow \min_{\mathbf{S}} & \mathrm{Tr}(\mathbf{S}^\top \mathbf{A} \mathbf{S}) - \mathrm{Tr}(\mathbf{S}^\top \mathbf{B}) \\ s.t. & \mathbf{S}\mathbf{1}_n = \mathbf{1}_n,\ \mathbf{S}_{ij} \geq 0,\ \mathbf{S}_{ii} = 0, \mathbf{S} = \mathbf{S}^\top. \end{array} \tag{9}$$

where $\mathbf{A} = \alpha \mathbf{L_C} + \beta \mathbf{I}_n$, $\mathbf{B} = \mathbf{K} + 2\beta \mathbf{C}$ and $\mathbf{K} = \mathbf{X}\mathbf{X}^\top$ is a kernel matrix of dataset $\mathbf{X}$. Then this problem can be solved using the Augmented Lagrangian Multiplier (ALM) method (Lin, Chen, and Ma 2010). To this end, we reformulate it as the following equivalent problem:

$$\begin{array}{ll} \min_{\mathbf{S}} & \mathrm{Tr}(\mathbf{S}^\top \mathbf{A} \mathbf{U}) - \mathrm{Tr}(\mathbf{S}^\top \mathbf{B}) \\ s.t. & \mathbf{S}\mathbf{1}_n = \mathbf{1}_n,\ \mathbf{S}_{ij} \geq 0,\ \mathbf{S} = \mathbf{U}, \mathbf{U}_{ii} = 0, \mathbf{U} = \mathbf{U}^\top, \end{array} \tag{10}$$

where $\mathbf{U}$ is an auxiliary variable. Then its corresponding augmented Lagrangian function is,

$$\mathcal{L} = \mathrm{Tr}[\mathbf{S}^\top (\mathbf{A}\mathbf{U} - \mathbf{B})] + \frac{\mu}{2} \|\mathbf{S} - \mathbf{U} + \frac{\mathbf{Y}}{\mu}\|_F^2, \tag{11}$$

where $\mathbf{Y}$ is the Lagrange multiplier and μ is a parameter that is adaptively updated.

A. Updating $\mathbf{S}$ with fixed other variables. Given fixed $\mathbf{U}$ and $\mathbf{Y}$, updating $\mathbf{S}$ involves minimizing $\mathcal{L}$, which is equivalent to solving the following optimization problem:

$$\begin{array}{ll} \min_{\mathbf{S}} & \|\mathbf{S} - \mathbf{M}\|_F^2 \\ s.t. & \mathbf{S}\mathbf{1}_n = \mathbf{1}_n,\ \mathbf{S}_{ij} \geq 0, \end{array} \tag{12}$$

where $\mathbf{M} = \mathbf{U} - (\mathbf{Y} + \mathbf{A}\mathbf{U} - \mathbf{B})/\mu$. Then the simplex projection method (Huang, Nie, and Huang 2015) can be used to update $\mathbf{S}$ row by row.

B. Updating $\mathbf{U}$ with fixed other variables. To update $\mathbf{U}$, we solve the following problem:

$$\begin{array}{ll} \min_{\mathbf{U}} & \|\mathbf{U} - \mathbf{N}\|_F^2 \\ s.t. & \mathbf{U} = \mathbf{U}^\top,\ \mathbf{U}_{ii} = 0, \end{array} \tag{13}$$

where $\mathbf{N} = \mathbf{S} + (\mathbf{Y} - \mathbf{A}^\top \mathbf{S})/\mu$. The closed-form solution is given by $\mathbf{U} = (\mathbf{N} + \mathbf{N}^\top)/2$, followed by setting the diagonal elements of $\mathbf{U}$ to zero.

C. Updating $\mathbf{Y}$ and μ with fixed other variables. The update rules for $\mathbf{Y}$ and μ are as follows:

$$\begin{array}{l} \mathbf{Y} \leftarrow \mathbf{Y} + \mu(\mathbf{S} - \mathbf{U}), \\ \mu \leftarrow \min(\rho\mu, \mu_{\max}), \end{array} \tag{14}$$

where $\rho = 1.5$ and $\mu_{\max} = 10^{30}$.

We use **Algorithm 1** to describe the algorithmic procedure for solving $\mathbf{S}$.

Algorithm 1: The ALM method

Input:
The matrix $\mathbf{A}$ and $\mathbf{B}$, the parameter α, β;
Output:
The affinity graph $\mathbf{S}$;
Initialize $\mathbf{S}$ by Eq.(4) ;
while not convergent **do**
  Update $\mathbf{S}$ by solving Problem (12);
  Update $\mathbf{U}$ by solving Problem (13);
  Update $\mathbf{Y}, \mu$ by Eq. (14);
end while

2. C-subproblem Suppose that $\mathbf{S}$ and $\mathbf{F}$ have been computed, Problem (8) is reduced as follows:

$$\begin{aligned} &\min_{\mathbf{C}} && \alpha \mathrm{Tr}(\mathbf{S}^\top \mathbf{L_C} \mathbf{S}) + \beta \|\mathbf{S} - \mathbf{C}\|_F^2 \\ &&& + \lambda \mathrm{Tr}(\mathbf{F}^\top \mathbf{L_C} \mathbf{F}) \\ &\Leftrightarrow \min_{\mathbf{C}} && -\alpha \mathrm{Tr}[\mathbf{C}^\top \mathbf{S}\mathbf{S}^\top) + \beta \|\mathbf{C} - \mathbf{S}\|_F^2 \\ &&& - \lambda \mathrm{Tr}(\mathbf{C}^\top \mathbf{F}\mathbf{F}^\top) \\ &\Leftrightarrow \min_{\mathbf{C}} && \mathrm{Tr}(\mathbf{C}^\top \mathbf{A}\mathbf{C}) - \mathrm{Tr}(\mathbf{C}^\top \mathbf{B}) \\ &s.t. && \mathbf{C}\mathbf{1}_n = \mathbf{1}_n,\ \mathbf{C}_{ij} \geq 0, \mathbf{C}_{ii} = 0, \mathbf{C} = \mathbf{C}^\top. \end{aligned} \tag{15}$$

Here, we let $\mathbf{A} = \beta \mathbf{I}_n$ and $\mathbf{B} = \alpha \mathbf{S}\mathbf{S}^\top + 2\beta \mathbf{S} + \lambda \mathbf{F}\mathbf{F}^\top$. This problem could also be solved using **Algorithm 1**.

3. F-subproblem When $\mathbf{S}$ and $\mathbf{C}$ have been updated, Problem (8) is transferred to the following problem:

$$\min_{\mathbf{F}} \mathrm{Tr}(\mathbf{F}^\top \mathbf{L_C} \mathbf{F}), \quad s.t. \quad \mathbf{F}^\top \mathbf{F} = \mathbf{I}_c. \tag{16}$$

Then $\mathbf{F}$ is composed of the eigenvectors corresponding to the c smallest eigenvalues of the matrix $\mathbf{L_C}$.

Algorithm

We finally summarize the optimization procedure for solving the SGR problem in **Algorithm 2**. After obtaining $\mathbf{S}$, we apply the Ncuts algorithm to derive the final clustering results. Although it is also possible to use $\mathbf{C}$ to get the clustering results, due to the presence of its hard-rank constraint, the results obtained from $\mathbf{C}$ are empirically shown—see subsequent experiments—to be inferior to those derived from $\mathbf{S}$.

Algorithm 2: Self-learned Graph Regression

Input:
The dataset $\mathbf{X}$, parameters α, β, the number of clusters c, the maximal number of iteration T;
Output:
The affinity graph $\mathbf{S}$ and $\mathbf{C}$;
Initialize $\mathbf{S}$ by using Eq. (4), and initialize $\mathbf{C} = (\mathbf{1}_n \mathbf{1}_n^\top - \mathbf{I}_n)/n$;
while not convergent and $t < T$ **do**
  $t = t + 1$;
  Update $\mathbf{S}$ by using Algorithm 1;
  Update $\mathbf{C}$ by using Algorithm 1;
  Update $\mathbf{F}$ by solving Problem (16);
end while

Notice: Because λ should be a relatively large value, we initialize $\lambda = 15$. And in each iteration, λ is updated by using the same strategy in CAN.

Further Discussions

Acceleration Technique and Computation Analysis

We begin by briefly analyzing the time complexity of **Algorithm 1**.

When solving the $\mathbf{S}$ subproblem using **Algorithm 1**, the simplex projection method is applied to update $\mathbf{S}$ row by row in each iteration (i.e., Problem (12)). This process requires computing the matrix $\mathbf{M} = \mathbf{U} - (\mathbf{Y} + \mathbf{A}\mathbf{U} - \mathbf{B})/\mu = \mathbf{U} - \big[\mathbf{Y} + (\alpha \mathbf{L_C} + \beta \mathbf{I}_n)\mathbf{U} - (\mathbf{K} + 2\beta \mathbf{C})\big]/\mu$. Assuming that $\mathbf{K} = \mathbf{X}\mathbf{X}^\top$ has been precomputed, the dominant computational cost arises from the matrix multiplication $\mathbf{L_C}\mathbf{U}$, which incurs a time complexity of $O(n^3)$. Therefore, the overall computational cost of **Algorithm 1** is at least $O(n^3)$, which also leads to high time complexity in **Algorithm 2**. To address this issue, we introduce an acceleration strategy to reduce the overall computational burden.

First, since $\mathbf{S}$ is ultimately required to be symmetric, imposing row-wise constraints is equivalent to applying column-wise constraints. Hence, we can equivalently update $\mathbf{S}$ column by column.

Second, because $\mathbf{S}$ is initialized using Eq. (4) with a predefined neighborhood size k (where $k \ll n$), we can record the indices of nonzero entries in each column of $\mathbf{S}$. During subsequent optimization, we aim to preserve this sparsity structure by updating only the recorded positions. In other words, we refines the values of the existing nonzero entries to improve the affinity graph, while keeping their positions fixed.

Third, assume that the indices of nonzero elements in the j-th column of $\mathbf{S}$ are denoted by $\mathbb{I}_j$. Then, the j-th column of $\mathbf{S}$ can be updated by solving the following optimization problem:

$$\begin{aligned} &\min_{\mathbf{S}(\mathbb{I}_j, j)} && \|\mathbf{S}(\mathbb{I}_j, j) - \mathbf{M}(\mathbb{I}_j, j)\|_2^2 \\ &s.t. && \mathbf{S}(\mathbb{I}_j, j)\mathbf{1}_k = 1, \mathbf{S}(i, j) \geq 0, \end{aligned} \tag{17}$$

where

$$\begin{aligned} \mathbf{M}(\mathbb{I}_j, j) &= \mathbf{U}(\mathbb{I}_j, j) - \big[\mathbf{Y}(\mathbb{I}_j, j) + \mathbf{A}(\mathbb{I}_j, \mathbb{I}_j)\mathbf{U}(\mathbb{I}_j, j) \\ &\quad - \mathbf{B}(\mathbb{I}_j, j)\big]/\mu, \\ \mathbf{A}(\mathbb{I}_j, \mathbb{I}_j) &= \alpha \mathbf{L_C}(\mathbb{I}_j, \mathbb{I}_j) + \beta \mathbf{I}_n(\mathbb{I}_j, \mathbb{I}_j), \\ \mathbf{B}(\mathbb{I}_j, j) &= \mathbf{K}(\mathbb{I}_j, j) + 2\beta \mathbf{C}(\mathbb{I}_j, j). \end{aligned}$$

One may question the justification for approximating the nonzero entries of the j-th column in $\mathbf{A}\mathbf{U}$ using $\mathbf{A}(\mathbb{I}_j, \mathbb{I}_j)\mathbf{U}(\mathbb{I}_j, j)$. This is reasonable because the ultimate constraint is $\mathbf{S} = \mathbf{U}$, implying that the nonzero positions in the j-th column of both $\mathbf{S}$ and $\mathbf{U}$ should coincide. Therefore, restricting computation to these positions is both valid and efficient.

Under this formulation, computing $\mathbf{M}(\mathbb{I}_j, j)$ incurs a cost of $O(k^2)$, while the simplex projection step costs $O(k \log k)$. Consequently, the total cost of updating $\mathbf{S}$ across all columns is reduced to $O(nk^2 + nk \log k)$.

On the other hand, updating $\mathbf{U}$ via Problem (13) requires computing $\mathbf{N} = \mathbf{S} + (\mathbf{Y} - \mathbf{A}\mathbf{S}^\top)/\mu$. Based on the previous analysis, $\mathbf{N}$ can also be computed in a column-wise manner. Specifically, for the j-th column, we have $\mathbf{N}(\mathbb{I}_j, j) = \mathbf{S}(\mathbb{I}_j, j) + [\mathbf{Y}(\mathbb{I}_j, j) - \mathbf{A}(\mathbb{I}_j, \mathbb{I}_j)\mathbf{S}^\top(\mathbb{I}_j, j)]/\mu$. The cost for

computing all columns of $\mathbf{N}$ is thus $O(nk^2)$. Since $\mathbf{U}$ is finally updated via $\mathbf{U} = \mathbf{N} + \mathbf{N}^\top$ with $\mathbf{U}_{ii} = 0$, the total cost becomes $O(n^2 + n + nk^2)$.

Additionally, updating the Lagrange multiplier $\mathbf{Y}$ requires $O(n^2)$ time. Considering that the number of iterations in **Algorithm 1** is $m \ll n$, the total time complexity for solving the $\mathbf{S}$ subproblem using **Algorithm 1** is approximately $O(n^2)$.

When updating $\mathbf{C}$, suppose that $\mathbf{F}\mathbf{F}^\top$ has already been computed. By employing a similar strategy as used for updating $\mathbf{S}$, we restrict the update of the j-th column of $\mathbf{C}$ to the entries indexed by $\mathbb{I}_j$. Under this constraint, the computational cost of solving the $\mathbf{C}$ subproblem remains $O(n^2)$.

To update $\mathbf{F}$, we compute the c eigenvectors corresponding to the c smallest eigenvalues of the Laplacian matrix $\mathbf{L_C}$, which requires $O(n^2)$ time ($c \ll n$).

In addition, as mentioned above, updating $\mathbf{S}$ and $\mathbf{C}$ requires the precomputation of $\mathbf{X}\mathbf{X}^\top$ and $\mathbf{F}\mathbf{F}^\top$, each of which incurs a cost of $O(n^2)$. Therefore, the overall time complexity of **Algorithm 2** is $O(ln^2) \approx O(n^2)$, where $l \ll n$ denotes the number of outer iterations.

Convergence Analysis

The proposed acceleration technique merely restricts the positions of the non-zero elements in $\mathbf{S}$ and $\mathbf{C}$ that are allowed to be updated. Consequently, it does not affect the convergence behavior of **Algorithm 2**. Then we analyze the convergence of **Algorithm 2**.

First, noting that $\mathbf{S}_{ij}, \mathbf{C}_{ij} \geq 0$, the objective function of the SGR model can be expressed as $\mathbf{J}(\mathbf{S}, \mathbf{C}, \mathbf{F}) = \mathrm{Tr}(\mathbf{X}^\top \mathbf{L_S} \mathbf{X}) + \alpha \mathrm{Tr}(\mathbf{S}^\top \mathbf{L_C} \mathbf{S}) + \beta \|\mathbf{S} - \mathbf{C}\|_F^2 + \lambda \mathrm{Tr}(\mathbf{F}^\top \mathbf{L_C} \mathbf{F}) = \sum_{i,j} \left(\frac{1}{2} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \mathbf{S}_{ij} + \frac{\alpha}{2} \|\mathbf{s}_i - \mathbf{s}_j\|_2^2 \mathbf{C}_{ij} + \lambda \|\mathbf{f}_i - \mathbf{f}_j\|_2^2 \mathbf{C}_{ij} \right) + \beta \|\mathbf{S} - \mathbf{C}\|_F^2 \geq 0$.

To update $\mathbf{S}$ and $\mathbf{C}$, we employ **Algorithm 1** (the Augmented Lagrangian Method), which is theoretically guaranteed to converge when optimizing with respect to two variable blocks. Therefore, at the t-th iteration, the following inequality holds: $\mathbf{J}(\mathbf{S}^t, \mathbf{C}^t, \mathbf{F}^{t-1}) \leq \mathbf{J}(\mathbf{S}^{t-1}, \mathbf{C}^{t-1}, \mathbf{F}^{t-1})$.

Furthermore, $\mathbf{F}$ is updated via eigen-decomposition, which yields the exact minimizer of the corresponding subproblem (i.e., Problem (16)). Therefore, at the same iteration, it also holds that $\mathbf{J}(\mathbf{S}^t, \mathbf{C}^t, \mathbf{F}^t) \leq \mathbf{J}(\mathbf{S}^{t-1}, \mathbf{C}^{t-1}, \mathbf{F}^{t-1})$.

Based on this monotonic decrease and the non-negativity of the objective function, we conclude that the proposed optimization algorithm for solving the SGR model is guaranteed to converge.

Moreover, it is important to note that the above conclusions are derived under the assumption that the hyper-parameters α, β, and λ remain fixed. However, during the optimization process, in order to enforce the constraint $rank(\mathbf{L_C}) = n - c$, λ may be adjusted dynamically—either increased or decreased (see the footnote on Page 2). As a result, the objective function $\mathbf{J}(\mathbf{S}, \mathbf{C}, \mathbf{F})$ may exhibit oscillations during the initial iterations. Nevertheless, once λ stabilizes after a few iterations, the value of the objective function begins to decrease monotonically.

Experiments

In this section, we conduct clustering experiments to compare the performance of SGR with various existing representative graph-based clustering algorithms.

Experiment Setup

Datasets Six datasets, including Zoo[3], JAFFE (Lyons, Budynek, and Akamatsu 1999), ORL (Samaria and Harter 1994), COIL20[4], COIL100[5]and MNIST[6], are used. The information of these datasets is summarized in Table 1.

Datasets	Samples	dimensionality	Clusters
Zoo	101	16	7
JAFFE	213	4096 (64 × 64)	10
ORL	400	4096 (64 × 64)	40
COIL20	1440	1024 (32 × 32)	20
COIL100	7200	1024 (32 × 32)	100
MNIST	10000	576 (24 × 24)	10

Table 1: The statistical information of the used datasets.

Compared Algorithms Eight representative graph-based clustering algorithms, including Ncuts (Shi and Malik 2000), CAN (Nie, Wang, and Huang 2014), CLR (Constrained Laplacian Rank) (Nie et al. 2016), AOPL (Adaptive Order Proximity Learning) (Wu et al. 2022), SGL (Structured Graph Learning) (Wu et al. 2023), HNAC (Hybrid Nongreedy Asynchronous Clustering) (Uykan 2023), MOGC (Multi-order Graph Clustering) (Liu et al. 2024), and SDSGC (Structured Doubly Stochastic Graph-based Clustering) (Wang et al. 2025), are used for comparison.

Evaluation Metrics Three widely used metrics, including clustering accuracy (ACC), normalized mutual information (NMI), and adjusted rand index (ARI), are used to evaluate the performance of these algorithms.

Parameters Setting There are three tunable hyper-parameters in SGR: α, β and k. Both α and β are selected from the set $\{0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01\}$. Note that relatively small values are chosen for these two parameters. This setting ensures that the second and third terms in the objective function of the SGR do not overly dominate optimization, thus helping $\mathbf{S}, \mathbf{C}$ to learn the structure information from the original datasets. The neighborhood size k is chosen as a relatively large value of 30, which could allow the SGR to find a satisfactory solution. We will further investigate the sensitivity of SGR to these hyper-parameters in subsequent subsections. For all compared methods, the hyper-parameters are set according to the optimal configurations reported in their original publications. The experiments are carried out on a PC with Intel i9-10900 K, 64 G RAM,

[3]https://archive-beta.ics.uci.edu/

[4]https://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php

[5]https://git-disl.github.io/GTDLBench/datasets/coil100_datasets

[6]http://www.cad.zju.edu.cn/home/dengcai/Data/MLData.html

Matlab R2023a and Windows 11. In addition, we fixed the random number seed to make the experimental results reproducible.

Methods	Datasets					
	Zoo	JAFFE	ORL	COIL20	COIL100	MNIST
	ACC					
Ncuts	65.83	87.40	53.62	78.97	58.16	51.64
CAN	61.39	96.71	53.50	83.96	56.39	27.62
CLR	60.39	96.71	46.75	80.69	45.37	56.22
AOPL	61.39	95.77	59.00	80.69	59.07	58.08
SGL	61.39	92.95	54.26	77.36	42.40	56.24
HNAC	64.98	93.78	56.34	79.64	66.21	56.63
MOGC	64.36	97.14	*59.62*	79.30	65.49	60.48
SDSGC	*67.33*	*98.12*	58.00	*84.72*	*68.79*	*63.84*
SGR-S	**82.18**	**99.53**	**67.75**	**98.33**	**82.79**	**68.21**
SGR-C	61.39	92.96	56.75	79.31	48.36	60.40
	NMI					
Ncuts	64.49	91.26	71.23	87.57	77.51	56.02
CAN	65.69	97.36	72.05	91.94	84.07	58.63
CLR	64.52	97.36	65.62	88.59	79.83	55.72
AOPL	68.89	95.23	74.14	88.24	79.46	60.24
SGL	66.47	91.70	65.62	83.15	66.36	57.34
HNAC	70.05	94.41	72.42	89.69	85.96	57.66
MOGC	68.01	97.36	75.03	89.06	84.29	61.97
SDSGC	71.52	*97.80*	74.25	*92.86*	*86.86*	62.37
SGR-S	**83.44**	**99.18**	**82.26**	**98.33**	**96.75**	**75.82**
SGR-C	*72.98*	89.90	*77.26*	86.83	76.11	*69.04*
	ARI					
Ncuts	42.62	81.82	18.26	69.08	42.94	47.87
CAN	48.44	93.12	19.75	78.86	30.26	39.64
CLR	47.24	93.12	11.96	71.44	29.89	44.29
AOPL	47.24	91.01	23.08	71.35	22.85	54.33
SGL	51.49	87.21	12.23	69.26	24.63	48.39
HNAC	54.43	95.92	21.18	72.38	49.28	52.67
MOGC	54.56	95.04	24.60	75.77	53.89	55.86
SDSGC	*56.89*	*96.89*	23.08	*79.13*	*55.08*	*60.89*
SGR-S	**81.75**	**98.98**	**55.27**	**96.74**	**81.61**	**61.11**
SGR-C	52.46	85.24	*25.59*	74.10	29.36	59.17

Table 2: The experimental results (in %) of SGR (highlighted with a grey background) and other evaluated algorithms. SGR-S and SGR-C denote the results obtained using $\mathbf{S}$ and $\mathbf{C}$, respectively, as generated by the SGR method. The best results are emphasized in bold, while the second-best results are underlined and emphasized in italics.

Experimental Results and Analyses

Comparison Results We first present the experimental results obtained by SGR and the compared algorithms in Table 2. From this table, we get the following observations:

1. SGR-S (i.e., SGR based on $\mathbf{S}$) consistently outperforms all competing methods across all datasets, demonstrating its robustness to different types of data. Notably, on several datasets such as Zoo, COIL20, and COIL100, the ACC achieved by SGR-S surpasses that of the second-best method by more than 10 percentage points. Similarly, the NMI and ARI scores of SGR-S on these datasets are also substantially higher than those of the second-best approach;

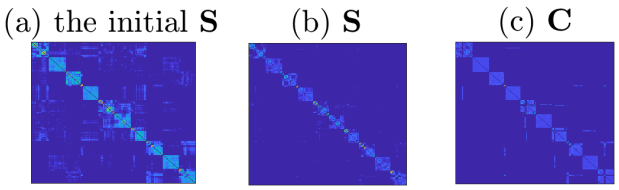

Figure 1: The two learned affinity matrices $\mathbf{S}$ and $\mathbf{C}$ of SGR and the initial value of $\mathbf{S}$. Zoom in for a better view.

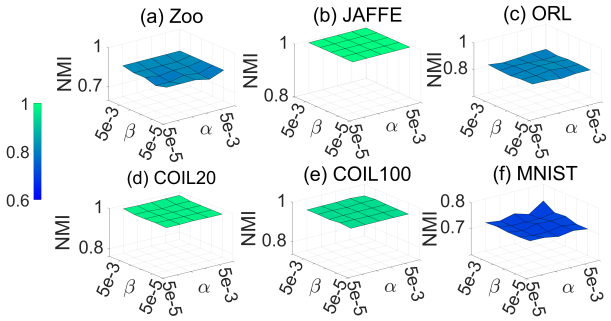

Figure 2: The performance of SGR-S changed with α, β. Here, k is fixed to be 30.

2. SGR-S also outperforms SGR-C. This performance gap may stem from the hard-rank constraint imposed on $\mathbf{C}$. To illustrate this, we visualize the two affinity graphs produced by SGR (as well as the initial value of $\mathbf{S}$) on the JAFFE dataset in Fig. 1. Based on this figure, one may argue that $\mathbf{C}$ exhibits a clearer block-diagonal structure than $\mathbf{S}$. However, it can be seen that $\mathbf{C}$ generates more connections between the samples from different classes. As a result, SGR-C achieves an ACC of 92.96%, whereas SGR-S reaches 99.53%.

Parameter Sensitivity We evaluate the sensitivity of SGR-S to its hyper-parameters using NMI as the representative performance metric. As shown in Figure 2, the performance of SGR-S is stable across different values of α and β within the predefined range. In contrast, Fig. 3 illustrates how the performance of SGR-S varies with k. Specifically, for each value of k ranging in $\{20, 30, 40, 50, n-1\}$[7], we record the optimal NMI achieved by SGR while tuning α and β, and then plot the resulting performance curve against k. It can be seen that the performance of SGR remains stable as k varies.

We also report the running times of SGR with $k = 30$ and $k = n - 1$, as well as SDSGC, in Fig. 4. It can be observed that while the runtime of SGR with $k = 30$ is slightly higher than that of SDSGC, it is significantly lower than that of SGR with $k = n - 1$.

These results demonstrate that the effectiveness of the proposed acceleration technique and SGR is robust to variations in all three hyper-parameters, highlighting its practical

[7] $n - 1$ means that for a data sample, all the other samples are considered as its neighbors. Namely, the accelerate strategy is not used.

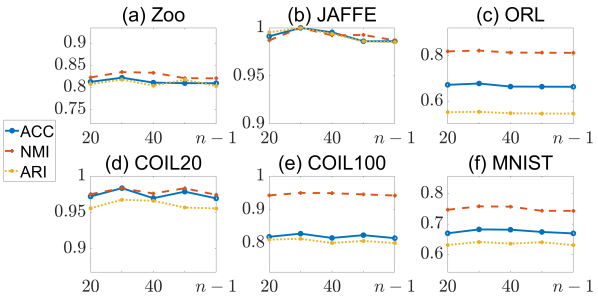

Figure 3: The performance of SGR-S changed with k. In each sub-figure, the horizontal axis indicates the value of k, and the vertical axis indicates values of different metrics.

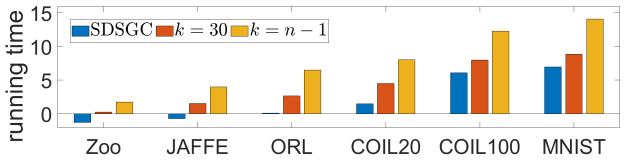

Figure 4: The running time (in logarithmic) of SGR with $k = 30$ and $k = n - 1$ as well as those of SDSGC.

utility in real-world applications.

Convergence We illustrate the convergence of Algorithm 2 in Fig. 5. It can be seen that the algorithm can always achieve convergence after 50 iterations. As discussed in the previous subsection, during the initial iterations, fluctuations in the value of λ may lead to non-monotonic changes in the objective function values. This phenomenon is noticeable in COIL20, COIL100 and MNIST datasets, as shown in Figures 5(d), (e) and (f). However, as the iterations progress and λ stabilizes, the objective function exhibits a clear monotonic decrease, confirming the convergence of **Algorithm 2**.

Ablation Study In this subsection, we address two potential concerns: (1) Is the self-learned graph regression really more effective than the k-nn-based graph regression? (2) Should the hard-rank constraint be imposed on $\mathbf{S}$?

To investigate the first concern, we formulate an alternative model (denoted as $\mathbf{P}_1$):

$$\begin{aligned} \min_{\mathbf{S}} \quad & \textstyle\sum_{i,j} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \mathbf{S}_{ij} + \alpha \mathrm{Tr}(\mathbf{S}^\top \mathbf{L}_{\mathbf{W}} \mathbf{S}) \\ s.t. \quad & \mathbf{S}\mathbf{1}_n = \mathbf{1}_n,\ \mathbf{S}_{ij} \geq 0, \mathbf{S}_{ii} = 0,\ \mathbf{S} = \mathbf{S}^\top, \\ & rank(\mathbf{L}_{\mathbf{S}}) = n - c, \end{aligned} \tag{18}$$

where $\mathbf{W}$ is a k-nn graph constructed by Eq. (4). The second concern gives rise to another variant (denoted as $\mathbf{P}_2$), which retains the full SGR model, but imposes the rank constraint on $\mathbf{L}_{\mathbf{S}}$ instead of $\mathbf{L}_{\mathbf{C}}$.

We conduct the same experiments as in the previous subsection to evaluate the performance of $\mathbf{P}_1$ and $\mathbf{P}_2$. The hyper-parameters used for $\mathbf{P}_2$ are kept identical to those in SGR. For $\mathbf{P}_1$, since it relies on a fixed k-nn graph $\mathbf{W}$ that inherently encodes structural information, we expand the search space of α to: $\{5\mathrm{e}{-5}, 1\mathrm{e}{-4}, 5\mathrm{e}{-4}, 1\mathrm{e}{-3}, 5\mathrm{e}{-3}, 1\mathrm{e}{-2}, 0.1, 1, 10, 50, 100, 200\}$. This expanded search range enables $\mathbf{P}_1$ to fully exploit the structural information embedded in $\mathbf{W}$. Finally, we report the experimental results in Table 3. From this table, we can see that SGR outperforms $\mathbf{P}_1$ and $\mathbf{P}_2$, proving the effectiveness of SGR and the rationality of the hard-rank constraint setting in SGR.

Methods	Datasets					
	Zoo	JAFFE	ORL	COIL20	COIL100	MNIST
	ACC					
$\mathbf{P}_1$	75.25	98.12	58.75	82.92	65.28	59.22
$\mathbf{P}_2$	77.23	98.12	65.50	87.36	60.72	34.49
SGR	**82.18**	**99.53**	**67.75**	**98.33**	**82.79**	**68.21**
	NMI					
$\mathbf{P}_1$	75.53	97.36	76.64	95.41	87.73	70.12
$\mathbf{P}_2$	79.03	97.36	81.51	95.41	84.50	52.49
SGR	**83.44**	**99.18**	**82.26**	**98.33**	**96.75**	**75.82**
	ARI					
$\mathbf{P}_1$	66.94	95.92	37.33	78.59	61.52	43.09
$\mathbf{P}_2$	80.08	95.92	37.07	82.48	52.82	19.61
SGR	**81.75**	**98.98**	**55.27**	**96.74**	**81.61**	**61.11**

Table 3: The experimental results (in %) of $\mathbf{P}_1$, $\mathbf{P}_2$ and SGR. For SGR and $\mathbf{P}_2$, we only report the results based on $\mathbf{S}$.

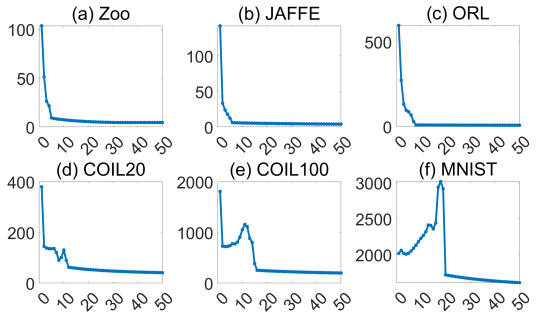

Figure 5: The objection values against the number of iterations. The vertical axis in each sub-figure indicate the objection value.

Conclusions

To address the limitations of existing graph-based clustering methods with locality constraints, we propose to leverage the affinity graph learned during optimization to construct its own graph regression. To facilitate efficient optimization, we relax the original formulation into a problem that is convex with respect to each variable, and present the Self-learned Graph Regression (SGR) model. We also design an acceleration strategy to further improve computational efficiency. Extensive experiments conducted on various benchmark datasets demonstrate the superior clustering performance of SGR. Furthermore, comprehensive ablation studies validate the effectiveness of the self-learned graph regression mechanism and provide insights into the impact on the design of the hard-rank constraint.

References

Bai, L.; Liang, J.; and Zhao, Y. 2023. Self-Constrained Spectral Clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(4): 5126–5138.

Bai, L.; Qi, M.; and Liang, J. 2023. Spectral clustering with robust self-learning constraints. *Artif. Intell.*, 320: 103924.

Chen, M.-S.; Huang, L.; Wang, C.-D.; Huang, D.; and Yu, P. S. 2022. Multiview Subspace Clustering With Grouping Effect. *IEEE Transactions on Cybernetics*, 52(8): 7655–7668.

Ding, M.; Yang, J.-H.; Zhao, X.-L.; Zhang, J.; and Ng, M. K. 2025. Nonconvex Low-Rank Tensor Representation for Multi-View Subspace Clustering with Insufficient Observed Samples. *IEEE Transactions on Knowledge and Data Engineering*, 1–14.

Elhamifar, E.; and Vidal, R. 2013. Sparse Subspace Clustering: Algorithm, Theory, and Applications. *IEEE Trans Pattern Anal Mach Intell*, 35(11): 2765–81.

Fu, J.; Cheng, K.; Song, A.; Xia, Y.; Chang, Z.; and Shen, Y. 2024. FSS-DBSCAN: Outsourced Private Density-Based Clustering via Function Secret Sharing. *IEEE Trans. Inf. Forensics Secur.*, 19: 7759–7773.

Huang, J.; Nie, F.; and Huang, H. 2015. A New Simplex Sparse Learning Model to Measure Data Similarity for Clustering. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, 3569–3575.

Ji, P.; Zhang, T.; Li, H.; Salzmann, M.; and Reid, I. D. 2017. Deep Subspace Clustering Networks. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 24–33.

Lee, H.-W.; Malik, N.; Shi, F.; and Mucha, P. J. 2019. Social clustering in epidemic spread on coevolving networks. *Phys. Rev. E*, 99: 062301.

Lei, T.; Jia, X.; Zhang, Y.; Liu, S.; Meng, H.; and Nandi, A. K. 2019. Superpixel-Based Fast Fuzzy C-Means Clustering for Color Image Segmentation. *IEEE Trans. Fuzzy Syst.*, 27(9): 1753–1766.

Lin, Z.; Chen, M.; and Ma, Y. 2010. The Augmented Lagrange Multiplier Method for Exact Recovery of Corrupted Low-Rank Matrices. *CoRR*, abs/1009.5055.

Liu, C.; Wu, Z.; Wen, J.; Xu, Y.; and Huang, C. 2023. Localized Sparse Incomplete Multi-View Clustering. *IEEE Trans. Multim.*, 25: 5539–5551.

Liu, G.; Lin, Z.; Yan, S.; Sun, J.; Yu, Y.; and Ma, Y. 2013. Robust Recovery of Subspace Structures by Low-Rank Representation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 35(1): 171–184.

Liu, Y.; Lin, X.; Chen, Y.; and Cheng, R. 2024. Multi-order graph clustering with adaptive node-level weight learning. *Pattern Recognit.*, 156: 110843.

Lu, C.; Feng, J.; Lin, Z.; Mei, T.; and Yan, S. 2019. Subspace Clustering by Block Diagonal Representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(2): 487–501.

Lyons, M. J.; Budynek, J.; and Akamatsu, S. 1999. Automatic Classification of Single Facial Images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(12): 1357–1362.

Nie, F.; Wang, X.; and Huang, H. 2014. Clustering and projected clustering with adaptive neighbors. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, 977–986.

Nie, F.; Wang, X.; Jordan, M. I.; and Huang, H. 2016. The Constrained Laplacian Rank Algorithm for Graph-Based Clustering. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, 1969–1976.

Nie, F.; Wu, D.; Wang, R.; and Li, X. 2020. Self-Weighted Clustering With Adaptive Neighbors. *IEEE Trans. Neural Networks Learn. Syst.*, 31(9): 3428–3441.

Patel, V. M.; Nguyen, H. V.; and Vidal, R. 2013. Latent Space Sparse Subspace Clustering. In *ICCV*, 225–232.

Pei, S.; Chen, H.; Nie, F.; Wang, R.; and Li, X. 2023. Centerless Clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(1): 167–181.

Samaria, F.; and Harter, A. 1994. Parameterisation of a stochastic model for human face identification. In *Proceedings of Second IEEE Workshop on Applications of Computer Vision, WACV 1994, Sarasota, FL, USA, December 5-7, 1994*, 138–142. IEEE.

Sarfraz, M. S.; Sharma, V.; and Stiefelhagen, R. 2019. Efficient Parameter-Free Clustering Using First Neighbor Relations. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, 8934–8943.

Shi, J.; and Malik, J. 2000. Normalized Cuts and Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8): 888–905.

Taştan, A.; Muma, M.; and Zoubir, A. M. 2024. Fast and Robust Sparsity-Aware Block Diagonal Representation. *IEEE Transactions on Signal Processing*, 72: 305–320.

Uykan, Z. 2023. Fusion of Centroid-Based Clustering With Graph Clustering: An Expectation-Maximization-Based Hybrid Clustering. *IEEE Trans. Neural Networks Learn. Syst.*, 34(8): 4068–4082.

Wang, J.; Ma, Z.; Nie, F.; and Li, X. 2022. Entropy regularization for unsupervised clustering with adaptive neighbors. *Pattern Recognit.*, 125: 108517.

Wang, N.; Cui, Z.; Li, A.; Lu, Y.; Wang, R.; and Nie, F. 2025. Structured Doubly Stochastic Graph-Based Clustering. *IEEE Transactions on Neural Networks and Learning Systems*, 1–14.

Wei, L.; Chen, Z.; Yin, J.; Zhu, C.; Zhou, R.; and Liu, J. 2023. Adaptive Graph Convolutional Subspace Clustering. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 6262–6271.

Wei, L.; Li, K.; Zhou, R.; and Liu, J. 2025. Purely Contrastive Multiview Subspace Clustering. *IEEE Transactions on Cybernetics*, 1–14.

Wei, L.; Liu, S.; Zhou, R.; Zhu, C.; and Liu, J. 2024. Learning Idempotent Representation for Subspace Clustering. *IEEE Transactions on Knowledge and Data Engineering*, 36(3): 1183–1197.

Wu, D.; Chang, W.; Lu, J.; Nie, F.; Wang, R.; and Li, X. 2022. Adaptive-order proximity learning for graph-based clustering. *Pattern Recognit.*, 126: 108550.

Wu, D.; Nie, F.; Lu, J.; Wang, R.; and Li, X. 2023. Effective Clustering via Structured Graph Learning. *IEEE Trans. Knowl. Data Eng.*, 35(8): 7909–7920.

Yin, M.; Gao, J.; and Lin, Z. 2016. Laplacian Regularized Low-Rank Representation and Its Applications. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(3): 504–517.

Zhang, X.; Ge, Y.; Qiao, Y.; and Li, H. 2021. Refining Pseudo Labels With Clustering Consensus Over Generations for Unsupervised Object Re-Identification. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, 3436–3445.