Scalable Attributed-Graph Subspace Clustering

Chakib Fettal[1,2], **Lazhar Labiod**[1], **Mohamed Nadif**[1]

[1] Centre Borelli UMR 9010, Université Paris Cité, 75006 Paris, France
[2] Informatique Caisse des Dépôts et Consignations
{firstname.lastname}@u-paris.fr

Abstract

Over recent years, graph convolutional networks emerged as powerful node clustering methods and have set state of the art results for this task. In this paper, we argue that some of these methods are unnecessarily complex and propose a node clustering model that is more scalable while being more effective. The proposed model uses Laplacian smoothing to learn an initial representation of the graph before applying an efficient self-expressive subspace clustering procedure. This is performed via learning a factored coefficient matrix. These factors are then embedded into a new feature space in such a way as to generate a valid affinity matrix (symmetric and non-negative) on which an implicit spectral clustering algorithm is performed. Experiments on several real-world attributed datasets demonstrate the cost-effective nature of our method with respect to the state of the art.

Introduction

An attributed-graph is a type of graph that contains two information sources, a topology or structure and node- and/or edge-level features. Under different approaches, they are used to model a wide variety of structured data (Fettal, Labiod, and Nadif 2023, 2022b), with applications in the fields of recommender systems (Fan et al. 2019; Ying et al. 2018), computer vision (Satorras and Estrach 2018; Yang et al. 2018), Natural language processing (Marcheggiani and Titov 2017) and physical systems (Hoshen 2017).

With the advent of the Graph Convolutional Network (GCN) (Defferrard, Bresson, and Vandergheynst 2016; Kipf and Welling 2016), graph related tasks such as graph representation learning (Wu et al. 2019; Zhu and Koniusz 2021) and graph clustering (Anton Tsitsulin and Müller 2020) have received a lot of attention. We observe, however, that for the task of graph clustering, few approaches (Cai et al. 2020) based on the subspace clustering principle have been proposed despite it being, at first sight, well-suited to attributed-graph data. We argue that this is mostly due to subspace clustering models suffering from high spatial and/or computational complexity. In a nutshell, the goal of subspace clustering is to group data points according to the subspaces in which they lie within a dataset. For example, subspace clustering models that use the self-expressive property, whereby every data point can be represented as an approximate linear combination of other points, have to learn a square matrix called the *coefficient* or *self-representation* matrix. This coefficient matrix has a size that is quadratic in the number of points. Once this matrix is learned, an affinity matrix is constructed from it and spectral clustering is performed on said affinity matrix. We can see the classical subspace clustering pipeline in figure 1.

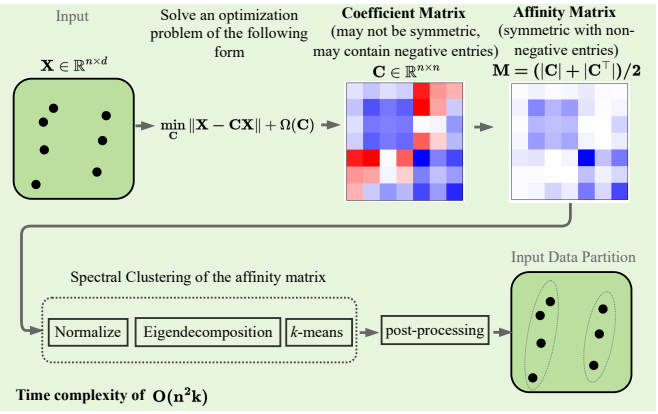


Figure 1: The traditional subspace clustering pipeline. A coefficient matrix $\mathbf{C}$ is initially learned. An affinity matrix $\mathbf{M}$ is then generated based on the magnitudes of $\mathbf{C}$, e.g., a common choice is $\mathbf{M} = (|\mathbf{C}| + |\mathbf{C}^\top|)/2$. Finally, a partition of the data is created via applying spectral clustering on $\mathbf{M}$.

In this paper, we argue that subspace clustering is well-suited to attributed-graph representations generated with GCN-based models due the neighborhood averaging making the data points closer and thus helping with the self-expressiveness of the data points. To leverage this property and in order to avoid the complexity problems associated with traditional subspace clustering, we propose an efficient variant to learn an initial representation of the graph before applying an efficient self-expressive subspace clustering procedure via learning a factored coefficient matrix and then projecting these factors into a new feature space in such a way as to generate a valid affinity matrix (symmetric with non-negative entries) on which to perform implicit spectral clustering. A schema for our model is available in figure

3. To showcase the efficacy and efficiency of our proposal, we perform extensive experimentation on six widely used attributed-networks. We can see a preview of the results in figure 2, these are the clustering results of our model on the Arxiv open graph benchmark, our model yields a 14% improvement over the second best model in terms of performance and 16% improvement in terms of speed. Code for our paper can be found in [1].

This paper is organized as follows: Section 2 reviews related works. Section 3 presents the necessary previous work. Section 4 is devoted to the proposed model and its computational complexity study. In section five, we carry out our experimental study. Finally we present our conclusion in section 6.

Related Work

Subspace Clustering

Subspace clustering methods based on the self-expressive property are commonly used on image data and have set state-of-the-art results on the task of image clustering. One of the earlier approaches was the Least-Square Regression subspace clustering (LSR) that leverages a grouping effect in the data. Newer models that make up the state-of-the-art include the Elastic-net Subspace Clustering (EnSC) (You et al. 2016) that uses a mix of l1- and l2-norm regularization, and the Subspace Clustering through the Orthogonal Matching Pursuit (SSC-OMP) (You, Robinson, and Vidal 2016) which possesses a subspace-preserving affinity under broad conditions. There are also deep learning approaches like the deep Subspace clustering network (Ji et al. 2017) and but these models have received some critique to the effect that their good performances are the result of an ad-hoc post processing step instead of the actual self-representation learning process (Haeffele, You, and Vidal 2021). More recently, a new efficiency trend has appeared, and some scalable models have also been proposed e.g. k-Factorization Subspace Clustering (k-FSC) (Fan 2021) which was put forward as a scalable subspace clustering model that factorizes data into subsets via structured sparsity.

Attributed-Graph Clustering

In this paper, attributed-graph clustering refers to the process of grouping nodes into clusters according to the graph topology and node features. We can classify attributed-graph clustering models into two subsets. A first one, where the goal is to learn graph representations and then use traditional clustering models such as k-means. Examples of models that use this approach include Simplified Graph Convolution (SGC) (Wu et al. 2019) which proposes a neighborhood averaging process that corresponds to a fixed low-pass filter, and the Simple Spectral Graph Convolution (S²GC) which uses a new method for the aggregation of K-hop neighborhoods that is a trade-off of low- and high-pass filter bands. (Zhu and Koniusz 2021). On the other hand, the second class of attributed-graph clustering models proposes to include the clustering objective into the representation learning process

[1] https://github.com/chakib401/sagsc

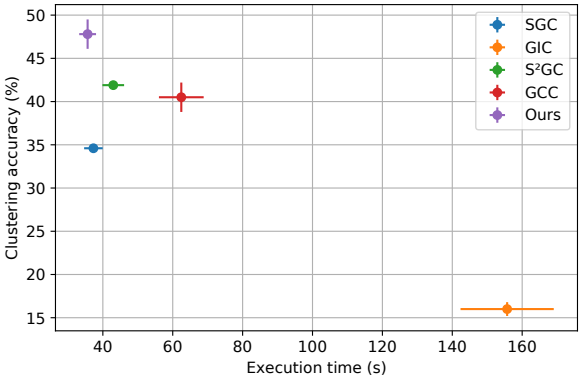


Figure 2: Clustering accuracy scores (%) plotted against the execution time (s) for our method and the state-of-the-art attributed-graph clustering models on the OGBN-arXiv dataset.

to learn better results, e.g., Graph InfoClust (GIC) (Mavromatis and Karypis 2021) which generates clusters by maximizing mutual information between nodes contained in the same cluster, and Graph Convolutional Clustering (GCC) (Fettal, Labiod, and Nadif 2022a) that performs clustering by minimizing the difference between convolved node representations and their reconstructed cluster representatives.

Preliminaries

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A}, \mathbf{X})$ be an undirected attributed-graph where $\mathcal{V}$ is the vertex set consisting of nodes $\{v_1, \ldots, v_n\}$, $\mathcal{E}$ is the set of edges that connects the nodes, $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a symmetric adjacency matrix where a_{ij} denotes the edge weight between nodes v_i and v_j, if $a_{ij} = 0$ then there is no edge between v_i and v_j, and $\mathbf{X} \in \mathbb{R}^{n \times d}$ is a node-level feature matrix. Our goal is to partition this graph into k independent subsets in an unsupervised manner.

Graph Convolutional Networks

The Graph Convolutional Network consists in a sequence of propagation layers. It can be formalized recursively as

$$\mathbf{H}^{(l+1)} \leftarrow \sigma\left(\hat{\mathbf{D}}^{-1/2}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-1/2}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right) \quad \text{with} \quad \mathbf{H}^{(0)} = \mathbf{X} \tag{1}$$

where $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix with added self-loop the, $\hat{\mathbf{D}}$ is its diagonal matrix of degrees, σ is some activation function and $\mathbf{W}^{(l)}$ is the weight matrix of the l-th layer. These weight matrices are optimized for some downstream task like semi-supervised classification, link prediction, etc.

Simplified Graph Convolutional Networks

Authors in (Wu et al. 2019) argued that the non-linearities in the GCN are superfluous and that most of its performance comes from the feature propagation. With this, the recursive

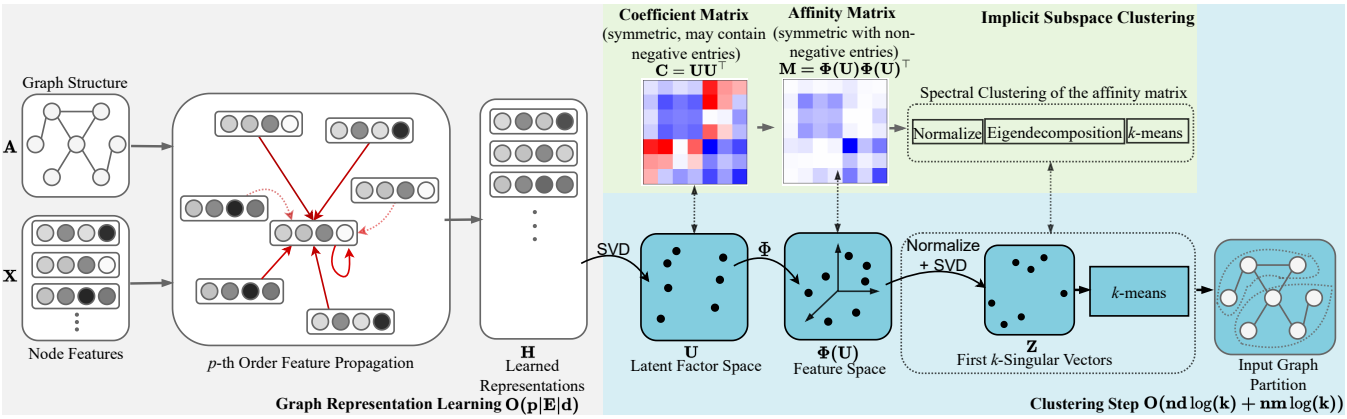


Figure 3: Diagram of our proposal. We have as input an attributed-graph characterized by an adjacency matrix $\mathbf{A}$ and a feature matrix $\mathbf{X}$. An initial representation $\mathbf{H}$ of the attributed-graph is learned through neighborhood propagation. Then, subspace clustering is performed using a latent factor matrix $\mathbf{U}$ where $\mathbf{C} = \mathbf{U}\mathbf{U}^\top$ is the subspace coefficient matrix that we project using a quadratic kernel feature map Φ so that $\mathbf{M} = \Phi(\mathbf{U})\Phi(\mathbf{U})^\top \geq 0$. With this we obtain the final partition by using the k-means algorithm on $\mathbf{Z}$, the first k singular vectors (not counting the first one) of $\mathbf{D}^{-1/2}\Phi(\mathbf{U})^\top$.

definition of a p-layer GCN collapses into

$$\mathbf{H} \leftarrow \mathbf{S}^p\mathbf{X}\mathbf{W}$$

where $\mathbf{S} = \hat{\mathbf{D}}^{-1/2}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-1/2}$ is called the propagation matrix. Here, we can see how the weights matrices collapsed into a single weight matrix $\mathbf{W}$ while the graph propagation steps collapsed into the p-th power of the propagation matrix $\mathbf{S}$.

Subspace Clustering

The goal of subspace clustering is to group data points according to the subspaces that support them. A popular formulation uses the self-expressive property where it is assumed that a data point can be written as a linear combination of the data points that belong to the same subspace. A possible formulation is

$$\min_{\mathbf{C}\in\mathcal{C}} \quad \|\mathbf{X} - \mathbf{C}\mathbf{X}\|^2 + \Omega(\mathbf{C}) \tag{2}$$

where $\mathbf{X} \in \mathbb{R}^{n\times d}$ is a matrix of d-dimensional data points, $\mathbf{C} \in \mathbb{R}^{n\times n}$ is known as the *self-representation* or *coefficient matrix*, $\Omega(\mathbf{C})$ is a regularization term introduced to establish certain properties for $\mathbf{C}$ e.g. to avoid trivial solutions (such as $\mathbf{C} = \mathbf{I}$), and $\mathcal{C}$ is the feasible region.

Once a solution $\mathbf{C}$ is found, an affinity matrix is generated based on the magnitudes of the entries of $\mathbf{C}$, a popular choice for this is $(|\mathbf{C}| + |\mathbf{C}^\top|)/2$. Finally, a clustering of the data points is obtained using some graph clustering algorithm such as the spectral clustering algorithm (Shi and Malik 2000).

Proposed Approach

In this paper, we propose the following generic formulation for the attributed-graph subspace clustering problem

$$\min_{\mathbf{C}\in\mathcal{C}} \quad \|\mathtt{agg}(\mathbf{A},\mathbf{X}) - \mathbf{C}\,\mathtt{agg}(\mathbf{A},\mathbf{X})\|^2 + \Omega(\mathbf{C}) \tag{3}$$

where $\mathtt{agg}$ is an aggregation function whose role is to merge the two information sources the topology information and the feature information present in the graph.

Simple Graph Convolutional Encoder

We propose to use a GCN-based encoder. More particularly, we use the convolution operation proposed in the simplified graph convolutional network along with the normalization of the adjacency matrix used in (Fettal, Labiod, and Nadif 2022a)

$$\min_{\mathbf{C}\in\mathcal{C}} \quad \|\mathbf{S}^p\mathbf{X} - \mathbf{C}\,\mathbf{S}^p\mathbf{X}\|^2 + \Omega(\mathbf{C}). \tag{4}$$

Now that we have our initial graph representation, we can present our clustering step.

Efficient Subspace Clustering

Learning the implicit coefficient matrix We set constraints on $\mathbf{C}$ in order to obtain a decomposition of $\mathbf{C}$ into the Gramian product $\mathbf{U}\mathbf{U}^\top$ where $\mathbf{U} \in \mathbb{R}^{n\times k}$ is a semi-orthogonal matrix i.e. $\mathbf{U}^\top\mathbf{U} = \mathbf{I}$. This will allow us to significantly speed up the subspace clustering process. Thus, our problem becomes

$$\min_{\mathbf{U}} \quad \|\mathbf{S}^p\mathbf{X} - \mathbf{U}\mathbf{U}^\top\,\mathbf{S}^p\mathbf{X}\|^2 \quad \text{such that} \quad \mathbf{U}^\top\mathbf{U} = \mathbf{I}. \tag{5}$$

As we can see, we have no need for any form of regularization. This problem can be efficiently solved through a singular value decomposition of the convolved features $\mathbf{S}^p\mathbf{X}$. With this we obtain a solution $\mathbf{C}$ which corresponds to a subspace coefficient matrix from which we can derive a clustering of the nodes.

Learning the implicit affinity matrix Once we have a coefficient matrix $\mathbf{C} = \mathbf{U}\mathbf{U}^\top$, we have to derive a non-negative matrix that reflects the magnitudes of the entries of $\mathbf{C}$. As already mentioned the common way is to compute $(|\mathbf{C}| + |\mathbf{C}^\top|)/2$ but this will result in a spectral clustering step which has a quadratic complexity in the number of nodes. In this paper, we propose to use a nonnegative kernel with feature map Φ to embed $\mathbf{U}$ into its feature space

explicitly

$$\mathbf{M} = \Phi(\mathbf{U})\Phi(\mathbf{U})^\top \geq 0. \tag{6}$$

Here the feature map is applied row-wise, for example, in our experiments, we used the quadratic kernel

$$\mathbf{M} = \mathbf{C}^{\circ 2} = (m_{ij}) = (c_{ij}^2). \tag{7}$$

It is also possible to introduce a bias term b to the kernel such as

$$m_{ij} = (c_{ij} + b)^2.$$

Hence, we have implicitly derived a Gramian decomposition through $\Phi(\mathbf{U})$ similarly to what was done for $\mathbf{C}$. This will allow us to efficiently perform the last step which corresponds to spectral clustering. Note that $\mathbf{M}$ is symmetric by construction.

Spectral clustering the implicit affinity matrix Through the previous step we can now efficiently perform the NJW spectral clustering (Ng, Jordan, and Weiss 2002) on matrix $\mathbf{M}$ by:

- Projecting the factor $\mathbf{U}$ using feature map Φ, i.e., $\mathbf{Q} \leftarrow \Phi(\mathbf{U})$
- Computing $\tilde{\mathbf{Q}} \leftarrow \mathbf{Q}\mathbf{D}^{-\frac{1}{2}}$ where $\mathbf{D}$ is a diagonal matrix such that d_{ii} is the sum of $\mathbf{M}$ i-th row.
- Constructing $\mathbf{Z}$ using the left singular vectors corresponding to the second to $k+1$-largest singular values of $\tilde{\mathbf{Q}}$.
- Performing a clustering of the rows of $\mathbf{Z}$ and assigning node i to cluster j if the i-th row of $\mathbf{Z}$ was assigned to cluster j.

Complexity Analysis

Our overall algorithm is presented in algorithm 1. In what follows, we will analyze the computational complexity of our proposal

Graph representation learning step To compute the p-th order graph convolution, we need $\mathcal{O}(p|E|d)$ operations.

Learning the implicit coefficient matrix Getting the left singular values of the convolved data requires $\mathcal{O}(nd\log(k))$ operations using the randomized singular value decomposition (Halko, Martinsson, and Tropp 2011).

Learning the implicit affinity matrix The projection of the data using a feature kernel of dimensionality m takes $\mathcal{O}(nm)$. The computation of the diagonal matrix $\mathbf{D}$ and its multiplication with $\mathbf{Q}$ takes $\mathcal{O}(nm)$ operations. The truncated singular value decomposition of $\hat{\mathbf{Q}}$ is in $\mathcal{O}(nm\log(k))$. Finally, the k-means algorithm applied on $\mathbf{Z}$ costs roughly $\mathcal{O}(nk^2)$. The overall computation time of this step $\mathcal{O}(nm\log(k) + nk^2)$.

Overall complexity. The totality of our algorithm cost $\mathcal{O}\left(p|E|d + n(m+d)\log(k) + nk^2\right)$. Generally, we have that $k << d$. The dimension m generally depends on k, for example in the case of the quadratic kernel $m = \binom{k+2}{2} = \frac{(k+2)(k+1)}{2}$. In other cases, when wishing to use nonnegative infinite dimensional kernels such as the RBF kernel or higher order polynomial kernels, feature map approximation techniques such as Nyströem method (Zhang, Tsang, and Kwok 2008) or the polynomial count sketch (Pham and Pagh 2013) can be used and m becomes a variable hyperparameter.

Algorithm 1: Scalable Attributed-Graph Subspace Clustering (SAGSC).

Input : $\mathbf{X}$ data matrix, $\mathbf{S}$ propagation matrix, p propagation order, k number of clusters, Φ nonnegative kernel feature map.
Output: π partition of the nodes.

$\mathbf{H} \leftarrow \mathbf{S}^p\mathbf{X}$;
Form the matrix $\mathbf{U}$ containing the first k left singular vectors of $\mathbf{H}$ in its rows;
$\mathbf{Q} \leftarrow \Phi(\mathbf{U})$;
$\mathbf{r} \leftarrow \mathbf{Q}^\top\mathbf{1}$;
$\mathbf{D} \leftarrow \mathtt{diag}(\mathbf{Q}\mathbf{r})$;
$\hat{\mathbf{Q}} \leftarrow \mathbf{Q}\mathbf{D}^{-\frac{1}{2}}$;
Form the matrix $\mathbf{Z}$ containing left singular vectors corresponding to the second to $k+1$-th largest singular values of $\hat{\mathbf{Q}}$ in its rows;
Apply a clustering algorithm on the rows of $\mathbf{Z}$ to obtain π a partition of the data;

In table 1, we can see how the complexity of our algorithm compares with that of the other models. Despite our model being using subspace clustering, it is significantly more efficient than the other subspace clustering models both in terms of computational and spatial complexity. When comparing with the SOTA attributed-graph clustering models, we can see that when $m \in \mathcal{O}(d)$ then our model has the same complexity as them. Which means that when taking a smaller m, e.g., $m \in \mathcal{O}(k)$, then our model should be more computationally efficient.

Experiments

In this section, we conduct experimentation to showcase the effectiveness and efficiency of our SAGSC model.

Datasets and Metrics

In our experiments, We use six commonly used benchmark datasets to compare the different models including three citation network datasets (ACM, DBLP (Wang et al. 2019); PubMed (Sen et al. 2008); and Wiki (Yang et al. 2015)), an Amazon sales dataset (Computers) (Shchur et al. 2018) and one large scale dataset (OGBN-arXiv) (Hu et al. 2020). The summary statistics of the datasets are shown in table 2.

We adopt three popular clustering evaluation metrics: clustering accuracy (CA), normalized mutual information (NMI) (Strehl and Ghosh 2002), adjusted rand index (ARI) (Hubert and Arabie 1985).

Baseline Models and Algorithms

The following are the baselines we used in our experiments:

- k**-Means** will serve as the simplest baseline.

Method	Time complexity	Space complexity
k-means	$\mathcal{O}(ndk)$	$\mathcal{O}(n(k+d))$
LSR	$\mathcal{O}(n^2k)$	$\mathcal{O}(n^2)$
EnSC	$\mathcal{O}(n^2k)$	$\mathcal{O}(n^2)$
SSC-OMP	$\mathcal{O}(n^2k)$	$\mathcal{O}(n^2)$
SGC	$\mathcal{O}(p\|E\|d+ndk)$	$\mathcal{O}(n(k+d))$
S²GC	$\mathcal{O}(p\|E\|d+ndk)$	$\mathcal{O}(n(k+d))$
GCC	$\mathcal{O}(p\|E\|d+ndk)$	$\mathcal{O}(n(k+d))$
SAGSC	$\mathcal{O}\left(p\|E\|d+n(d+m)\log(k)+nk^2\right)$	$\mathcal{O}\left(n(k+d+m)\right)$

Table 1: Complexity of the different models. For k-FSC, m refers to the dimension of subspaces. For k-FSC, many possible complexities are possible depending on the chosen algorithm, please see (Fan 2021) for a discussion on its complexity. For simplicity, we suppose that the embedding dimension in SGC, S²GC and GCC is in $\mathcal{O}(k)$.

Dataset	Nodes	Edges	Features	Classes	Imbalance
ACM	3025	16,153	1870	3	1.1
Wiki	2405	14,001	4973	17	45.1
DBLP	4057	2,502,276	334	4	1.6
Amazon Computers	13,381	259,159	767	10	17.5
Pubmed	19,717	64,041	500	3	1.9
OGBN-arXiv	169,343	1,327,142	128	40	942.1

Table 2: The datasets statistics. The imbalance is quantified via the ratio between the majority and minority classes.

- **LSR** is a subspace clustering model with an l2-norm regularization.
- **EnSC** is a subspace clustering model with an elastic net regularization (mix of l1- and l2-norm regularization).
- **SSC-OMP** has a subspace-preserving affinity under broad conditions.
- **k-FSC** is a scalable subspace clustering model that factorizes model in subsets via structured sparsity.
- **SC** refers to the classical spectral clustering algorithm applied on the original adjacency matrix of the graph.
- **SGC** proposes a neighborhood averaging process that corresponds to a fixed low-pass filter.
- **GIC** generates clusters by maximizing mutual information between nodes contained in the same cluster.
- **S²GC** proposes a new method for the aggregation of K-hop neighborhoods that is a trade-off of low- and high-pass filter bands.
- **GCC** performs clustering by minimizing the difference between convolved node representations and their reconstructed cluster representatives.

We use the implementations of the authors when possible.

Experimental Settings

All experiments were implemented in TensorFlow and conducted on a standard computer with a 12GB memory GPU an a RAM of 12GB. In all experiments, we ran the models ten times, and report the average performance along with the corresponding standard deviation. We use the implementations of the authors when possible but optimized them to run on GPU. We used hyper-parameters prescribed by authors when possible. For fairness, for the remaining hyper-parameters, we ran grid searches and reported the results corresponding to the best accuracy for all models. For k-FSC, we use the LARGE implementation. All results are the averages of ten runs.

For our model, we use a quadratic kernel feature map with a bias term equal to $\frac{1}{\sqrt{2}}$. This leads to the following kernel feature map:

$$\begin{aligned}\varphi : \mathbb{R}^k &\to \mathbb{R}^{\binom{k+2}{2}} \\ \mathbf{x} &\mapsto \langle x_k^2, \ldots, x_1^2, x_k x_{k-1}, \ldots, x_k x_1, x_{k-1}x_{k-2}, \\ &\ldots, x_{k-1}x_1, \ldots, x_2 z_1, x_k, \ldots, x_1, \tfrac{1}{\sqrt{2}}\rangle\end{aligned} \quad (8)$$

for the power hyper-parameter, similarly to the other benchmarks, we use a grid search over the accuracy and report the best results. We do however propose a heuristic to adaptively select this hyper-parameter.

Node Clustering Results

Performance Clustering performances of the different methods are reported in tables 3 and 4. Best performances are highlighted in bold while second best results are underlined. In table 3, there is a general trend that the methods that use both $\mathbf{A}$ and $\mathbf{X}$ perform better than those that use $\mathbf{A}$ or $\mathbf{X}$ individually, except on DBLP where they perform well. Our model has the best performance over the three datasets with respect to all three metrics. The GCC has the second best results in all but one case, i.e., ARI on Wiki where it is outperformed by S²GC. In table 4, the three datasets are of larger sizes, our model has the best results in eight out of nine

Method	Input	ACM			DBLP			Wiki		
		CA	NMI	ARI	CA	NMI	ARI	CA	NMI	ARI
k-means	**X**	87.8 ± 0.9	61.7 ± 1.5	67.4 ± 2.1	67.9 ± 0.0	37.3 ± 0.0	31.5 ± 0.1	47.6 ± 1.4	48.6 ± 0.2	26.6 ± 0.2
LSR	**X**	78.6 ± 0.0	43.1 ± 0.0	48.3 ± 0.0	69.4 ± 0.1	34.7 ± 0.1	36.4 ± 0.2	17.8 ± 0.5	2.8 ± 1.7	0.3 ± 0.2
EnSC	**X**	83.8 ± 0.0	53.0 ± 0.0	58.6 ± 0.0	30.0 ± 0.1	0.8 ± 0.2	0.1 ± 0.0	47.5 ± 0.0	45.2 ± 0.2	30.2 ± 0.1
SSC-OMP	**X**	82.1 ± 0.0	49.4 ± 0.1	55.3 ± 0.0	29.4 ± 0.1	0.4 ± 0.1	-0.1 ± 0.0	37.8 ± 8.5	34.4 ± 9.1	21.2 ± 7.9
k-FSC	**X**	59.7 ± 7.2	25.2 ± 7.1	27.2 ± 7.2	51.3 ± 11.1	17.4 ± 7.3	17.3 ± 9.6	38.2 ± 5.1	35.6 ± 3.9	17.7 ± 4.4
SC	**A**	36.5 ± 0.2	1.0 ± 0.2	0.7 ± 0.1	91.0 ± 0.0	73.0 ± 0.1	78.3 ± 0.1	30.7 ± 1.1	24.0 ± 0.8	6.0 ± 0.2
SGC	**A**, **X**	83.7 ± 0.0	55.7 ± 0.0	58.8 ± 0.0	88.8 ± 0.0	69.5 ± 0.0	73.2 ± 0.0	51.9 ± 0.8	49.6 ± 0.2	28.6 ± 0.1
GIC	**A**, **X**	90.1 ± 0.3	68.2 ± 0.6	73.2 ± 0.6	90.2 ± 0.2	72.4 ± 0.4	77.4 ± 0.3	48.0 ± 0.7	48.4 ± 0.3	31.0 ± 0.3
S²GC	**A**, **X**	84.1 ± 0.1	56.8 ± 0.1	59.6 ± 0.2	88.3 ± 0.0	69.2 ± 0.0	71.9 ± 0.0	52.1 ± 1.0	52.2 ± 0.1	33.0 ± 0.4
GCC	**A**, **X**	91.3 ± 0.0	71.2 ± 0.1	76.0 ± 0.1	91.8 ± 0.0	74.5 ± 0.0	80.5 ± 0.0	53.7 ± 1.4	**53.5 ± 0.5**	31.6 ± 1.1
SAGSC	**A**, **X**	**93.3 ± 0.1**	**75.1 ± 0.2**	**80.9 ± 0.1**	**93.1 ± 0.1**	**78.1 ± 0.2**	**83.2 ± 0.2**	**56.0 ± 2.1**	**53.5 ± 1.2**	**34.1 ± 2.7**

Table 3: Clustering performance of the different models over ACM, DBLP and Wiki. Best results are highlighted in bold font and second best results are underlined.

Method	Input	Amazon Computers			PubMed			OGBN-arXiv		
		CA	NMI	ARI	CA	NMI	ARI	CA	NMI	ARI
SGC	**A**, **X**	65.5 ± 0.0	52.2 ± 0.0	45.7 ± 0.0	69.6 ± 0.0	29.3 ± 0.0	29.9 ± 0.0	34.6 ± 0.4	39.2 ± 0.1	25.2 ± 0.6
GIC	**A**, **X**	46.8 ± 2.2	47.5 ± 0.9	31.3 ± 3.5	64.5 ± 0.4	26.2 ± 0.3	23.8 ± 0.4	16.0 ± 0.8	17.9 ± 0.5	5.8 ± 0.2
S²GC	**A**, **X**	65.4 ± 0.0	55.4 ± 0.0	**49.5 ± 0.0**	71.0 ± 0.0	**32.9 ± 0.0**	33.7 ± 0.0	41.9 ± 0.3	45.9 ± 0.1	36.9 ± 0.5
GCC	**A**, **X**	67.6 ± 0.0	56.0 ± 0.0	46.5 ± 0.0	70.5 ± 0.0	32.2 ± 0.0	33.1 ± 0.0	40.5 ± 1.7	46.8 ± 0.2	35.1 ± 2.0
SAGSC	**A**, **X**	**69.0 ± 1.0**	**58.2 ± 0.4**	48.2 ± 1.8	**71.1 ± 0.0**	**32.9 ± 0.0**	**34.1 ± 0.0**	**47.8 ± 1.7**	**47.1 ± 0.5**	**38.4 ± 1.6**

Table 4: Clustering performance of the SOTA models over the larger networks; Amazon Computers, Pubmed and OGBN-arXiv. Best results are highlighted in bold font and second best results are underlined.

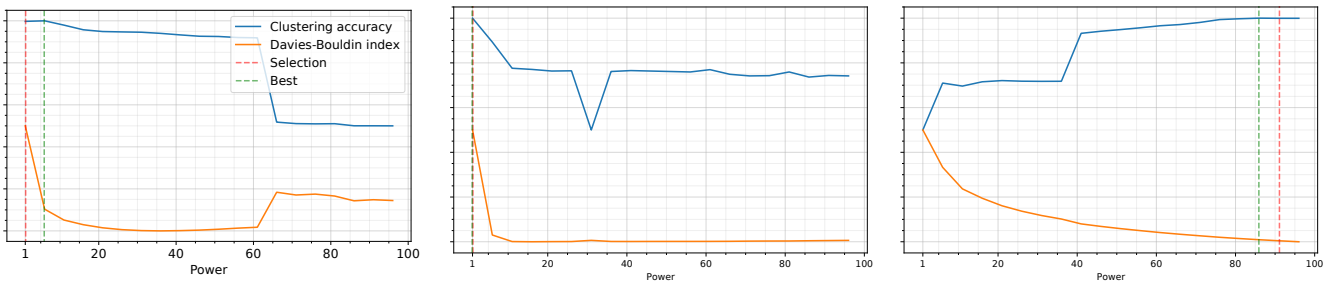
Method	ACM	DBLP	Wiki	Pubmed	Computers	OGBN-arXiv
k-means	4.29 ± 0.7	6.05 ± 0.7	24.25 ± 0.3	-	-	-
LSR	20.14 ± 0.26	5.2 ± 0.64	46.55 ± 1.61	-	-	-
EnSC	590.31 ± 63.93	120.66 ± 0.28	232.58 ± 3.04	-	-	-
SSC-OMP	201.78 ± 34.8	37.1 ± 2.87	293.78 ± 9.63	-	-	-
k-FSC	3.72 ± 0.87	8.45 ± 0.73	34.29 ± 1.86	-	-	-
SC	2.15 ± 0.32	18.54 ± 1.30	2.86 ± 0.44	-	-	-
SGC	0.56 ± 0.13	0.19 ± 0.04	1.68 ± 0.14	1.18 ± 0.39	1.00 ± 0.11	37.30 ± 2.66
GIC	3.67 ± 0.16	268.96 ± 63.17	5.96 ± 0.66	12.0 ± 1.50	22.38 ± 1.51	155.7 ± 13.34
S²GC	0.44 ± 0.04	0.23 ± 0.06	1.56 ± 0.10	0.82 ± 0.10	1.33 ± 0.28	42.98 ± 3.10
GCC	1.73 ± 2.96	0.33 ± 0.10	1.66 ± 0.14	1.26 ± 0.11	1.92 ± 0.13	62.45 ± 6.40
SAGSC	**0.40 ± 0.05**	**0.18 ± 0.05**	**1.07 ± 0.09**	**0.79 ± 0.05**	**0.88 ± 0.12**	**35.64 ± 2.41**

Table 5: Execution time of all methods in seconds. Best results are highlighted in bold.

cases. Our model has the second best result on the remaining case (ARI over Amazon Computers). Note that for NMI over PubMed, we have a tie between S²GC and our model. On the largest dataset OGBN-arXiv, our model shows a 14% improvement over the second best model, S²GC.

Efficiency In table 5, we report the training times of all the baselines over ACM, DBLP and Wiki, and report those of the SOTA over PubMed, Computers and OGBN-arXiv. Our model is the fastest one on all datasets. Two main observations can be made. First, our model is significantly faster than other subspace clustering models including the more efficient ones like k-FSC. Second, our model is as fast the SOTA attributed-graph clustering models despite it being based on subspace clustering which is know to be computationally heavy.

Analysis Overall, our model is as fast as the fastest attributed-graph clustering models while consistently yielding the best overall performance on all six datasets. This shows the cost-effective nature of our model with respect to the state of the art.



(a) ACM. (b) DBLP. (c) Pubmed.

Figure 4: Plot of the clustering accuracy (%) and the Davies-Bouldin index (Davies and Bouldin 1979) against the propagation power.

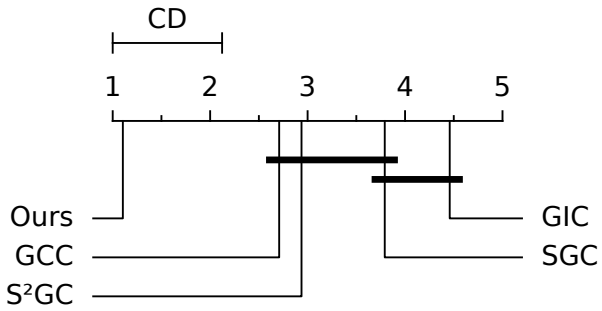


Figure 5: Results of the Nemenyi test where each rank represents the average rank over the CA, NMI, ARI and clustering F1-score; on the six datasets. We see that our model achieves the best rank, and is alone in the best performing group. We can also see the formation of two other groups.

To back up this claim statistically, we use the Nemenyi post-hoc test (Nemenyi 1963) to find groups of models that perform similarly in a statistically meaningful manner, to do this we rank the performances of the different models w.r.t four metrics (CA, NMI, ARI, and clustering F1-score) for each dataset. This yields 24 different rankings. We then carry-out the test with a confidence level of 90%. Results are illustrated in figure 5. We see the formation of three groups. The first one containing the best performing model, SAGSC; a second one, containing GCC, S²GC and SGC; and a third one containing SGC and GIC.

Selection of the Power Hyper-Parameter

The selection of the power parameter is integral to the performance of our model. A power that is too small can lead to not enough neighborhood information being propagated and a power that is too large can lead to the oversmoothing phenomenon (Chen et al. 2020). Since in the unsupervised context, it impossible to know for certain which power will lead to the best performance, several heuristics for the selection of this hyper-parameter have been proposed, e.g., in (Zhang et al. 2019), authors proposed to use internal criteria based on the information intrinsic to the data while in (Fettal, Labiod, and Nadif 2022a,b) authors proposed to choose a cutoff threshold on the change of their loss function between successive powers. Here, we propose to use an approach similar to the elbow method (Ketchen and Shook 1996) which is used for the selection of number of clusters in the k-means algorithm. We start by choosing an interval for the powers we wish to consider e.g. the multiples of five plus one between one and a hundred i.e. $\{1, 6, \ldots, 96\}$. Then we choose the power that precedes the appearance of the first pronounced 'elbow' in the graph. if there is no elbow, we choose the upper bound of the interval.

For example, in figure 4, we have can see a clear elbow for ACM, DBLP when the power is equal to six so we the power to one. In the case of Pubmed, no such elbow appears and so we set the power 96. We see that with very simple rule, we reach an accuracy that almost the same as the best one. For DBLP, we retrieve the best power, while for ACM and PubMed, the differences between the accuracy of the power we retrieved and the best one are 0.23 and 0.02, respectively, which is negligible. Of course, after this initial selection, a more granular selection needs can be performed since here we used an interval with a crude spacing of five between consecutive powers. Note that this selection process can be easily automated.

Conclusion

In this paper, we leveraged subspace clustering for attributed-graphs through the means of an efficient algorithm whereby after learning an initial representation of the graph through a simple yet effective neighborhood propagation step. We learn a factored coefficient matrix through orthogonal constraints, these factors are then embedded into a new feature space in such a way as to create a symmetric and nonnegative affinity matrix on which an implicit spectral clustering algorithm is performed. We additionally showed how this overall clustering process corresponds to an implicit subspace clustering algorithm. The experimentation we conducted showed the effectiveness and efficiency of our proposal with respect to the state of the art attributed-graph clustering algorithms.

References

Anton Tsitsulin, B. P., John Palowitch; and Müller, E. 2020. Graph Clustering with Graph Neural Networks. In *Proceedings of the 16th International Workshop on Mining and Learning with Graphs (MLG)*.

Cai, Y.; Zhang, Z.; Cai, Z.; Liu, X.; Jiang, X.; and Yan, Q. 2020. Graph convolutional subspace clustering: A robust subspace clustering framework for hyperspectral image. *IEEE Transactions on Geoscience and Remote Sensing*, 59(5): 4191–4202.

Chen, D.; Lin, Y.; Li, W.; Li, P.; Zhou, J.; and Sun, X. 2020. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 3438–3445.

Davies, D. L.; and Bouldin, D. W. 1979. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2): 224–227.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29: 3844–3852.

Fan, J. 2021. Large-Scale Subspace Clustering via k-Factorization. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 342–352.

Fan, W.; Ma, Y.; Li, Q.; He, Y.; Zhao, E.; Tang, J.; and Yin, D. 2019. Graph Neural Networks for Social Recommendation. In *The World Wide Web Conference*, WWW '19, 417–426. New York, NY, USA: Association for Computing Machinery. ISBN 9781450366748.

Fettal, C.; Labiod, L.; and Nadif, M. 2022a. Efficient Graph Convolution for Joint Node Representation Learning and Clustering. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, 289–297.

Fettal, C.; Labiod, L.; and Nadif, M. 2022b. Subspace Coclustering with Two-Way Graph Convolution. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 3938–3942.

Fettal, C.; Labiod, L.; and Nadif, M. 2023. Simultaneous Linear Multi-view Attributed Graph Representation Learning and Clustering. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*.

Haeffele, B. D.; You, C.; and Vidal, R. 2021. A Critique of Self-Expressive Deep Subspace Clustering. In *International Conference on Learning Representations*.

Halko, N.; Martinsson, P.-G.; and Tropp, J. A. 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2): 217–288.

Hoshen, Y. 2017. VAIN: Attentional Multi-agent Predictive Modeling. In *Neural Information Processing Systems (NIPS)*, 2701–2711.

Hu, W.; Fey, M.; Zitnik, M.; Dong, Y.; Ren, H.; Liu, B.; Catasta, M.; and Leskovec, J. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33: 22118–22133.

Hubert, L.; and Arabie, P. 1985. Comparing partitions. *Journal of classification*, 2(1): 193–218.

Ji, P.; Zhang, T.; Li, H.; Salzmann, M.; and Reid, I. 2017. Deep subspace clustering networks. *Advances in neural information processing systems*, 30.

Ketchen, D. J.; and Shook, C. L. 1996. The application of cluster analysis in strategic management research: an analysis and critique. *Strategic management journal*, 17(6): 441–458.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Marcheggiani, D.; and Titov, I. 2017. Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 1506–1515. Copenhagen, Denmark: Association for Computational Linguistics.

Mavromatis, C.; and Karypis, G. 2021. Graph InfoClust: Maximizing Coarse-Grain Mutual Information in Graphs. In *PAKDD (1)*, 541–553.

Nemenyi, P. B. 1963. *Distribution-free multiple comparisons.* Princeton University.

Ng, A. Y.; Jordan, M. I.; and Weiss, Y. 2002. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, 849–856.

Pham, N.; and Pagh, R. 2013. Fast and scalable polynomial kernels via explicit feature maps. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 239–247.

Satorras, V. G.; and Estrach, J. B. 2018. Few-Shot Learning with Graph Neural Networks. In *International Conference on Learning Representations*.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective classification in network data. *AI magazine*, 29(3): 93–93.

Shchur, O.; Mumme, M.; Bojchevski, A.; and Günnemann, S. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*.

Shi, J.; and Malik, J. 2000. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8): 888–905.

Strehl, A.; and Ghosh, J. 2002. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec): 583–617.

Wang, X.; Ji, H.; Shi, C.; Wang, B.; Ye, Y.; Cui, P.; and Yu, P. S. 2019. Heterogeneous graph attention network. In *The world wide web conference*, 2022–2032.

Wu, F.; Souza, A.; Zhang, T.; Fifty, C.; Yu, T.; and Weinberger, K. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*, 6861–6871.

Yang, C.; Liu, Z.; Zhao, D.; Sun, M.; and Chang, E. Y. 2015. Network Representation Learning with Rich Text Information. In *IJCAI*.

Yang, J.; Lu, J.; Lee, S.; Batra, D.; and Parikh, D. 2018. Graph R-CNN for Scene Graph Generation. In *Computer Vision - ECCV 2018 - 15th European Conference*, volume 11205 of *Lecture Notes in Computer Science*, 690–706.

Ying, R.; He, R.; Chen, K.; Eksombatchai, P.; Hamilton, W. L.; and Leskovec, J. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 974–983.

You, C.; Li, C.-G.; Robinson, D. P.; and Vidal, R. 2016. Oracle based active set algorithm for scalable elastic net subspace clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3928–3937.

You, C.; Robinson, D.; and Vidal, R. 2016. Scalable sparse subspace clustering by orthogonal matching pursuit. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3918–3927.

Zhang, K.; Tsang, I. W.; and Kwok, J. T. 2008. Improved Nyström low-rank approximation and error analysis. In *Proceedings of the 25th international conference on Machine learning*, 1232–1239.

Zhang, X.; Liu, H.; Li, Q.; and Wu, X.-M. 2019. Attributed Graph Clustering via Adaptive Graph Convolution. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, 4327–4333. International Joint Conferences on Artificial Intelligence Organization.

Zhu, H.; and Koniusz, P. 2021. Simple Spectral Graph Convolution. In *9th International Conference on Learning Representations, ICLR, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.