Self-Supervised Clustering based on Manifold Learning and Graph Convolutional Networks

Leonardo Tadeu Lopes
State University of São Paulo (UNESP), Brazil
leonardo.lopes@unesp.br

Daniel Carlos Guimarães Pedronette
State University of São Paulo (UNESP), Brazil
daniel.pedronette@unesp.br

Abstract

In spite of the huge advances in supervised learning, the common requirement for extensive labeled datasets represents a severe bottleneck. In this scenario, other learning paradigms capable of addressing the challenge associated with the scarcity of labeled data represent a relevant alternative solution. This paper presents a novel clustering method called Self-Supervised Graph Convolutional Clustering (SGCC)[1], which aims to exploit the strengths of different learning paradigms, combining unsupervised, semi-supervised, and self-supervised perspectives. An unsupervised manifold learning algorithm based on hypergraphs and ranking information is used to provide more effective and global similarity information. The hypergraph structures allow identifying representative items for each cluster, which are used to derive a set of small but high-confident clusters. Such clusters are taken as soft-labels for training a Graph Convolutional Network (GCN) in a semi-supervised classification task. Once trained in a self-supervised setting, the GCN is used to predict the cluster of remaining items. The proposed SGCC method was evaluated both in image and citation networks datasets and compared with classic and recent clustering methods, obtaining high-effective results in all scenarios.

1. Introduction

The amount of digital data generated every day is constantly growing, mainly supported by the technological advances in several areas of knowledge [4]. This huge amount of content creates an imperative demand for methods able to automatically organize, separate, and obtain knowledge from data. During the last decade, impressive advances has been achieved in supervised learning tasks, where labeled data is available for training machine learning models later used to make predictions. However, despite the substantial advances especially supported by the use of deep learning techniques [19], some challenges still remains, since such methods require extensive labeled sets for training. Furthermore, real-world scenarios often lack cataloged information. By demanding huge human efforts associated to its creation, the labeled sets are usually sparse, inaccurate or even nonexistent [40].

In order to overcome such limitations, research efforts have been put on other learning paradigms, capable of handling the scarcity of labeled data. In this scenario, several semi-supervised, unsupervised and self-supervised learning approaches have been recently proposed. Although some of them represent traditional research directions, they have attracted a crescent interest due to recent conditions and challenges. Semi-supervised learning techniques address the existence of a reduced number of labeled samples, mainly by exploiting abundant unlabeled data and expanding the information through the datasets [9, 35, 13, 10]. Unsupervised learning methods aim to obtain knowledge from data by only exploiting the relation between objects, without any labeled information [33, 11]. Self-supervised learning methods are another recent approach in which supervised or semi-supervised techniques are trained with knowledge obtained in an unsupervised manner, by pairwise comparisons or based on soft labels obtained from the original data [40].

In *unsupervised learning*, *clustering* is an important category of methods applied in several domains, such as computer vision, pattern recognition and data mining. Despite being a traditional field, it remains an active research area [14, 33, 24]. The main objective consists in discovering and separating the natural groupings of a set of patterns by exploring some inherent similarities between them [33]. Analogous to other machine learning techniques, clustering is highly dependent on the features and similarity measurement to achieve effective results: the better the similarity given by the representation embedded in the input feature-space, the better the groups found by those methods [24].

In a close direction to clustering, unsupervised *manifold learning* techniques exploit the structure of datasets to obtain better similarity relationships between the data samples. The methods take into account the dataset manifold through more global and effective similarity measures. Among the different approaches applied in manifold learning, ranked-based techniques have been established as a promising solution. They analyze the similarity relationship encoded in ranked lists, which provide a rich source of similarity information, to compute new similarity measures based on global data relationships [29]. Those new measures are used to improve retrieval and machine learning tasks [1, 30].

In *semi-supervised learning*, graph-based algorithms are widely exploited to extract the underlying structure of data. A promising research direction is given by the use of *Graph Convolutional Networks (GCNs)* [35], which have achieved state-of-the-art results on semi-supervised learning tasks. Such networks allow the learning of more effective embeddings through convolutional operations on non-Euclidean domains, defined by graph structures.

Moreover, GCNs have been recently applied to clustering tasks, exploiting relationships represented as graphs. In

[1]The code is available at https://github.com/lopes-leonardo/sgcc

[5], a clustering network exploits the relation between the node features and the normalized adjacency matrix using a Messaging Passing (MP) layer. The output is assigned to cluster by a Multi-Layer Perceptron (MLP) and the model is trained using a combined loss. A different approach is used in [7], where a *GCN* is combined with an autoencoder network to clusterize the dataset. Firstly, the autoencoder learns to reconstruct the input data in an unsupervised manner. The *GCN* layers receive the representation learned by the respective layer of the autoencoder, before the activation function. The framework is trained with a self-supervised approach based on the similarity between each object's autoencoder representation and its cluster center vector.

In this paper, we propose a novel clustering method called *Self-Supervised Graph Convolutional Clustering (SGCC)*. The strengths of different learning paradigms are exploited, combining unsupervised, semi-supervised and self-supervised perspectives. Firstly, the proposed approach employs an unsupervised manifold learning algorithm based on hypergraphs [29] to provide more effective and global similarity information. The hypergraph structures are exploited to select representative elements of the dataset and create a reliable initial cluster configuration. The initial clusters are modeled as soft-labels for training a GCN, used for classification in a semi-supervised manner. Subsequently, the GCN is used to predict the cluster of remaining data samples, resulting in a self-supervised clustering method. Furthermore, by processing the data at a feature level, our proposed approach can be applied to several domains, such as citation networks.

To the best of our knowledge, it is the first attempt that uses a robust similarity formulation given by hypergraph-based manifold learning to derive a self-supervised method based on GCNs for clustering tasks. Another relevant contribution is given by a novel approach for defining representative proxy elements and creating initial reliable clusters used for training the GCN models.

The effectiveness of the proposed method was assessed though a extensive experimental evaluation. Experiments were conduced on several datasets, including image datasets and citation networks, various GCN models, different evaluation measures, and comparison with classic and recent clustering techniques. A visual experiment was also conducted to demonstrate the ability of our method in projecting elements from the same classes closely in the embedding space computed by GCNs. The results obtained by the proposed approach were better or comparable with the considered methods in all evaluated scenarios.

2. Formal Definition

This section presents a formal definition for the three main tasks involved in the proposed approach.

2.1. Rank-Based Manifold Learning

Let $\mathcal{C} = \{o_1, o_2, \dots, o_n\}$ be a data collection, where each object o_i denotes a data item. Let $\mathbf{x}_i$ be a feature vector defined in $\mathbb{R}^d$, which represents the element $o_i \in \mathcal{C}$ in a d-dimensional feature space. The feature vectors are widely used for retrieval and machine learning tasks, commonly supported on distance or similarity measures computed between pairs of objects. Formally, let ρ: $\mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^+$ be a distance function that computes the distance between two objects based on their feature vectors. The distance between two objects o_i and o_j can be defined by $\rho(\mathbf{x}_i, \mathbf{x}_j)$. The traditional Euclidean distance is often employed. However, considering only pairs of objects can waste relevant information contained in more general relationships. In this scenario, ranked-based techniques aim to represent and exploit rich contextual similarity information.

A ranked list τ_q can be computed, based on the distance function ρ, to obtain the most similar objects of a given element o_q. Therefore, τ_q=(o_1, o_2, $\dots$, o_l) can be formally defined as a permutation of the collection $\mathcal{C}_l$, where l defines the length of the ranked list and $\mathcal{C}_l \subset \mathcal{C}$ is a sub-set containing the l most similar objects to o_q. The permutation τ_q is a bijection from the set $\mathcal{C}_l$ onto the set $[L_g] = \{1, 2, \dots, l\}$. Moreover, $\tau_q(o_i)$ represent the position of the object o_i in the ranked list τ_q. If o_i is ranked before o_j in the ranked list of o_q, that is, $\tau_q(o_i) < \tau_q(o_j)$, then $\rho(\mathbf{x}_q, \mathbf{x}_i) \leq \rho(\mathbf{x}_q, \mathbf{x}_j)$.

By computing a ranked list τ_i for every object $o_i \in \mathcal{C}$, the set $\mathcal{T}$ = $\{\tau_1, \tau_2, \dots, \tau_n\}$ of ranked lists can be obtained. This set encodes important similarity information, which takes into account the structure of the dataset. Rank-based manifold learning algorithms exploit the similarity information encoded in the set of ranked lists $\mathcal{T}$ to compute a new similarity measure, which in turn can be used to update the set of ranked lists. Formally, we can define an unsupervised manifold learning method as a function $m(\dot{)}$ which computes a more effective set of ranked lists $\mathcal{T}'$, as follows:

$$\mathcal{T}' = m(\mathcal{T}). \tag{1}$$

In this work, a hypergraph formulation [29] is used to instantiate the function $m(\cdot)$. The set $\mathcal{T}'$ and the hypergraph structures are used by the proposed clustering approach.

2.2. Clustering

Clustering can be defined as an unsupervised learning task that aims to extract natural disjunct groups from a data collection $\mathcal{C}$ [14]. Formally, let $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_c\}$ be a set of c clusters, where $\mathcal{S}_1 \bigcap \mathcal{S}_2 \bigcap \dots \bigcap \mathcal{S}_c = \varnothing$ and $\mathcal{S}_1 \bigcup \mathcal{S}_2 \bigcup \dots \bigcup \mathcal{S}_c = \mathcal{C}$. Every object $o_i \in \mathcal{C}$ is assigned to exactly one cluster $\mathcal{S}_j \in \mathcal{S}$ and does not belong to any other cluster [33].

2.3. Semi-Supervised Classification through GCNs

Recently, *GCNs* have been successfully applied to semi-supervised learning using graph-based representations of data [35]. We formally define semi-supervised classification tasks based on GCNs, according to [15]. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ be an undirected graph. Let $\mathcal{V} = \{v_1, v_2, \dots, v_n\}$ denotes a the set of graph nodes, where each $v_i \in \mathcal{V}$ represents an object of data collection $o_i \in \mathcal{C}$. Let $\mathcal{E}$ be the edge set and $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n]^T \in \mathbb{R}^{n \times d}$ be a feature matrix, where each $\mathbf{x}_i \in \mathbf{X}$ is a d-dimensional feature vector which represents the object o_i and its corresponding node v_i. The edge set, composed by a set of pairs $(v_i, v_j) \in \mathcal{E}$, can be defined by a non-negative adjacency matrix $\mathbf{A} = [a_{ij}] \in \mathbb{R}^{n \times n}$.

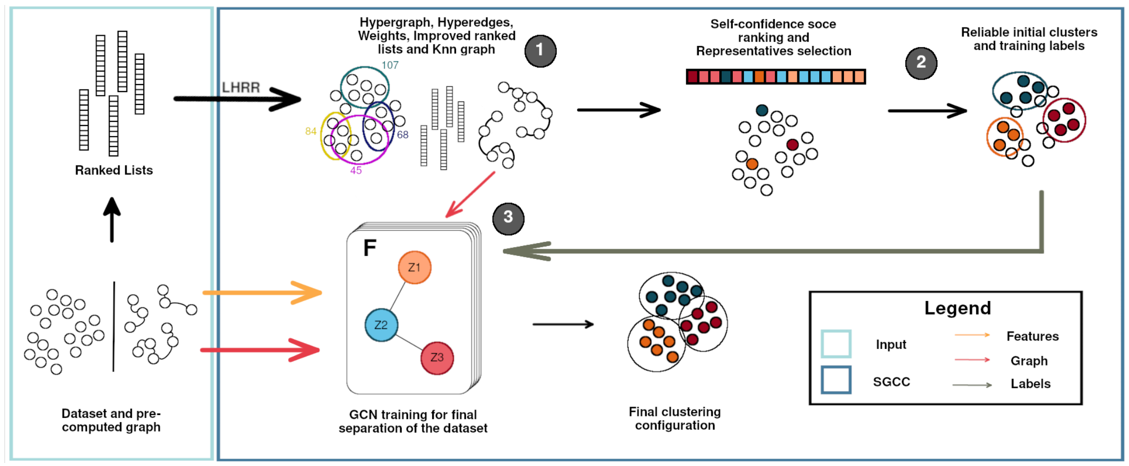
Ranked Lists
Dataset and pre-computed graph
LHRR
Hypergraph, Hyperedges, Weights, Improved ranked lists and Knn graph
1
107
84
68
45
Self-confidence soce ranking and Representatives selection
2
Reliable initial clusters and training labels
3
F
Z1
Z2
Z3
GCN training for final separation of the dataset
Final clustering configuration
Legend
Input
SGCC
Features
Graph
Labels

Figure 1. General structure and data flow of the proposed *Self-Supervised Graph Convolutional Clustering* (SGCC) method. Based on features and ranked lists from the original datasets, alongside with graph representations (when available), the proposed approach separates the data into clusters in three stages.

In addition, let $\mathcal{Y} = \{y_1, y_2, \dots, y_c\}$ be a set of labels, representing the target classes that can be assigned to nodes $v_i \in \mathcal{V}$. In this scenario, a partially labeled dataset can be defined as $\mathcal{V} = \{v_1, v_2, \dots, v_L, v_{L+1}, \dots, v_n\}$, where $\mathcal{V}_L = \{v_i\}_{i=1}^{L}$ represents a subset of labeled objects $\mathcal{V}_L \in \mathcal{V}$ and $\mathcal{V}_U = \{x_i\}_{i=L+1}^{n}$ is the subset of unlabeled objects $\mathcal{V}_U \in \mathcal{V}$. As a general rule, in semi-supervised classification, $|\mathcal{V}_L| \ll |\mathcal{V}_U|$. The training set can be defined by a function $l_b : \mathcal{V}_L \rightarrow \mathcal{Y}$, which assign labels such that $y_j = l_b(v_i)\ \forall\ v_i \in \mathcal{V}_L$. The main objective of GCN classification is to learn a new labeling function $\hat{l_b} : \mathcal{V}_U \rightarrow \mathcal{Y}$, which will predict labels of unlabeled nodes in $\mathcal{V}_U$.

3. Proposed Approach

This section presents the proposed self-supervised clustering approach, named *Self-Supervised Graph Convolutional Clustering (SGCC)*. The method is based on three main techniques, as illustrated in Figure 1 and discussed in detail in the following:

1. Manifold Learning based on Hypergraphs: A rank-based manifold learning algorithm [29] based on a hypergraph formulation is used to redefine the similarity among data samples. Additionally, the hypergraph structures are exploited by our approach in order to derive the initial clusters. The manifold algorithm is discussed in Section 3.1;

2. Self-Supervised Clustering: The more effective similarity and the hypergraph structures are used to derive a novel approach capable of identifying representative elements and creating small but reliable clusters. Section 3.2 describes how our method select representative elements and uses them to derive high-effective small clusters;

3. Graph Convolutional Network: A GCN is trained using the reliable clusters as soft-labels, using the original dataset features. A reciprocal k-NN graph is created from the improved ranked lists computed by the manifold learning algorithm, if the input data does not provide graph information. The GCN is used in a semi-supervised learning setting, exploiting both soft-labeled and unlabeled data. Section 3.3 describes the general concepts of GCNs and some of its variants used.

3.1. Manifold Learning based on Hypergraphs

The *Log-based Hypergraph of Ranking References (LHRR)* [29] is an unsupervised manifold learning method which computes more effective similarities among data elements. The method is based on ranking information from the set of ranked lists $\mathcal{T}$ modeled in hypergraph structures. The algorithm can be broadly divided in five main steps, which are described in the following sections:

3.1.1 Rank Normalization

Firstly, the method computes a new similarity measure p_n by using reciprocal rank positions. The p_n similarity between x_i and x_j can be defined as:

$$\rho_n(o_i, o_j) = 2l - (\tau_i(o_j) + \tau_j(o_i)), \tag{2}$$

where l denotes the ranked list length. The top-l items of the ranked lists are then updated by a stable sorting algorithm, based on the new computed similarities.

3.1.2 Hypergraph Construction

Hypergraphs are a robust generalization of graphs, where hyperedges can connect any set of vertices. Let $G = (V, E, w)$ be a hypergraph composed by a finite set of vertices V and by a hyperedge set E. Each item $o_i \in \mathcal{C}$ is associated to a vertice $v_i \in V$. The hyperedge set E can be defined as a family of subsets of V, such that $\bigcup_{e \in E} = V$. Additionally, a weight $w(e_i)$ is assigned to each hyperedge e_i, representing the confidence established by its relationships. A hyperedge e_i can be defined by a set of vertices $e_i = \{v_1, v_2, \dots, v_m\}$. In this configuration, a hyperedge e_i is incident with a vertex v_j when $v_j \in e_i$. Following this definition, we can represent the hypergraph using an incidence matrix $\mathbf{H}$ of size $|E| \times |V|$, such that:

$$h(e_i, v_j) = \begin{cases} r(e_i, v_j) & \text{if } v_j \in e_i, \\ 0 & \text{otherwise.} \end{cases},$$

where $r : E \times V \rightarrow \mathbb{R}^{+}$ is a membership function which indicates the degree of a vertex v_j belongs to a hyperedge e_i. The function exploits a second-order neighborhood relationship. For each data object $o_i \in \mathcal{C}$, a hyperedge e_i is defined based on the k-neighborhood set of o_i and neighbors of neighbors. Formally, the function $r(e_i, v_j)$ is defined as:

$$r(e_i, v_j) = \sum_{o_z \in \mathcal{N}(o_i,k) \wedge o_j \in \mathcal{N}(o_z,k)} w_p(o_i, o_z) \times w_p(o_z, o_j), \tag{3}$$

where $w_p(o_i, o_z)$ computes a relevance weight to o_z based on its position in the ranked list of o_i, namely τ_i. The function is computed through a log-based formulation:

$$w_p(o_i, o_z) = 1 - \log_k \tau_i(o_z). \tag{4}$$

Additionally, the Hyperedge Weight $w(e_i)$ measures the confidence of the relationships between the objects in hyperedge e_i. In order to compute $w(e_i)$, a Hypergraph Neighborhood Set $\mathcal{N}_h$ is defined, containing the k vertices with the highest scores $h(e_i, .)$ in the hyperedge. Considering the Hypergraph Neighborhood $\mathcal{N}_h$, an effective hyperedge is expected to contain few vertices with high values of $h(e_i, .)$ [29]. Therefore, the hyperedge weight $w(e_i)$ is computed as follows:

$$w(e_i) = \sum_{j \in \mathcal{N}_h(o_i,k)} h(e_i, v_j). \tag{5}$$

Moreover, since each hyperedge $e_i \in E$ is created based on the respective ranked list $\tau_i \in \mathcal{T}$, the Hyperedge Weight $w(e_i)$ can be considered as an unsupervised effectiveness measure of τ_i. Consequently, the more effective the ranked list, higher the hyperedge weight and more reliable the relationships contained in the hyperedge [29].

3.1.3 Hyperedge Similarities

After the hypergraph construction, LHRR [29] creates a new similarity matrix $\mathbf{S}_p$ based on two hypotheses. Firstly, similar objects have similar ranked lists and thus similar hyperedges. The similarity measure between two hyperedges e_i and e_j is given by its inner products, by multiplying the incidence matrix by its transposed: $\mathbf{S}_h = \mathbf{H}\mathbf{H}^T$.

The second hypothesis states that similar objects are expected to be referenced by the same hyperedges. Consequently, the similarity between two vertices v_i and v_j can be computed by multiplying the h values in their corresponding hyperedges, given by: $\mathbf{S}_v = \mathbf{H}^T\mathbf{H}$. Finally, the two similarities are combined by the Hadamard product between matrices $\mathbf{S}_h$ and $\mathbf{S}_v$, obtaining the similarity matrix $\mathbf{S}_p = \mathbf{S}_h \circ \mathbf{S}_v$.

3.1.4 Cartesian Product of Hyperedge Elements

Aiming to maximize the similarity information extracted from the hypergraph, LHRR exploits the vertices pairwise relationship by computing the Cartesian product between their respective hyperedges. Therefore, the Cartesian product between two hyperedges e_q and e_i can be defined as:

$$e_q \times e_i = \{(v_y, v_z) : v_y \in e_q \wedge v_z \in e_i\} \tag{6}$$

The Cartesian product can also be used to exploit the relationship between elements of the same hyperedge e_q, being described as e_q^2. For each pair $(v_i, v_j) \in {e_q}^2$, a relationship $p : E \times V \times V \rightarrow \mathbb{R}^+$ is established. This relationship is computed based on the weight $w(e_q)$, which represents the confidence of the hyperedge. Moreover, the relationship p also defines a membership degree between the vertices v_i and v_j with respect to e_q:

$$p(e_q, v_i, v_j) = w(e_q) \times h(e_q, v_i) \times h(e_q, v_j). \tag{7}$$

Based on the membership degree p, the matrix $\mathbf{Q}$ can be constructed considering the relationships across all hyperedges:

$$\mathbf{Q}_{i,j} = \sum_{e_q \in E \wedge (v_i,v_j) \in {e_q}^2} p(v_i, v_j). \tag{8}$$

3.1.5 Hypergraph-Based Similarity

Both matrices $\mathbf{S}_p$ and $\mathbf{Q}$ are combined to compute the Hypergraph-based similarity matrix $\mathbf{W} = \mathbf{Q} \circ \mathbf{S}_p$. This new similarity matrix, which concentrates all similarity information extracted from the hypergraph, is used to compute a new set of ranked lists for the data collection. Additionally, by using both input and output as ranked lists, the LHRR method can be iteratively repeated in order to obtain increasingly effective rankings and hypergraph representations. Let the superscript $^{(t)}$ denotes the current iteration, the set of ranked lists $\mathcal{T}^{(t+1)}$ is computed based on the similarity $\mathbf{W}^{(t)}$ and set $\mathcal{T}^{(0)}$ is defined by the initial feature representation.

For its application in SGCC, the LHRR method is repeated along t iterations. The final ranked list set $\mathcal{T}^{(t+1)}$, the current hyperedges set $E^{(t)}$ and its respective weights are used to initially cluster the input data, as discussed in the following sub-section.

3.2. Self-Supervised Clustering

This section discusses how the proposed SGCC method exploits the similarity information encoded into the hypergraph structures to create the initial reliable clusters. Subsequently, such clusters and the set of ranked lists computed by LHRR are used by GCNs for clustering remaining data items in a self-supervised manner. The self-supervised clustering approach, illustrated in Figure 1, and can be summarized in four main steps, discussed in the following:

1. Hyperedge Self-Confidence Score: A new score is computed in order to estimate the quality of similarity information encoded in the hyperedges. The hyperedges are ranked based on this estimation, such that top hyperedges present higher hyperedge weights and stronger similarity relationships with its neighbors;

2. Representatives Proxy Selection: Based on this score, a representative data item is selected for each of the desired clusters. These representatives items provide a high-effective initial representation for each cluster and are used for directing an agglomeration process;

3. Reliable Clusters Set: Each cluster is represented by a cluster hyperedge, initially defined by its respective representative's hyperedge. Subsequently, a sub-set of the data objects is agglomerated into the created clusters following the rank defined by the top hyperedge self-confidence scores. At each step, the next most reliable item is agglomerated based on its hyperedge relationship with each of the clusters. During the agglomeration, the cluster's hyperedge is updated by merging the new item's hyperedge;

4. Graph Convolution Network Clustering: Finally, based on the ranked lists retrieved from LHRR, a reciprocal k-NN graph is created. This new graph is combined with the dataset's feature set in order to train a GCN using the labels defined by reliable clusters. After training, the full dataset is classified by the graph-based network, retrieving a final clusters set.

Each step is further discussed and formally defined in the following sections.

3.2.1 Hyperedge Self-Confidence Score

As discussed in Section 3.1.2, the hyperedge weight $w(e_i)$ is an unsupervised effectiveness estimation of the ranked list τ_i, associated with the hyperedge e_i. Additionally, the incidence matrix score of a vertex v_i in its own hyperedge e_i, represented by $h(e_i, v_i)$, is incremented every time the o_i is referenced in the ranked list of an element present in $\mathcal{N}_h(o_i, k)$ and can be interpreted as an estimation of the reciprocal relationships contained in o_i's explored neighborhood. Aiming to obtain a general ranking of the most reliable items in the dataset, those two scores are combined in a *hyperedge self-confidence score* w_h:

$$w_h(e_i) = h(e_i, v_i) \times w(e_i). \tag{9}$$

Based on w_h, the ranked list τ_h=(o_1, o_2, ..., o_n) is defined as a permutation of the collection $\mathcal{C}$, such that if o_i is ranked before o_j, then $w_h(e_i) \geq w_h(e_j)$. The ranked list τ_h defines the order in which the dataset items are processed by the proposed algorithm. The more reliable items are selected and agglomerated earlier.

3.2.2 Representatives Proxy Selection

In spite of containing a reliable order of the dataset items according its similarity information confidence, τ_h can include various items similar to each other at top positions. In this scenario, inspired by [2], a selection of one representative element for each cluster is proposed in order to ensure diversity. These representatives will direct the creation of a reliable set of initial clusters.

Let $\mathcal{R} = (o_1, o_2, \ldots, o_c) \in \mathcal{C}$ be the set of selected representatives objects, such that $|\mathcal{R}| = c$ and each $o_i \in \mathcal{R}$ is selected by the following equation:

$$o_i = \underset{o_j \,\in\, \mathcal{C} \,\setminus\, \mathcal{R}}{\arg\max} \; \frac{w_h(o_j)}{1 + \sum_{o_k \,\in\, \mathcal{R}_{i-1}} h(e_k, v_j)}, \tag{10}$$

where $\mathcal{R}_{i-1} = (o_1, \ldots, o_{i-1})$ represents the set of previously selected representatives. Equation 10 can be summarized as: select the next candidate with a high self-confidence score (numerator) and low similarity with representatives selected in previous iterations (denominator).

The representatives set $\mathcal{R}$ is initialized with the first element in the ranked list τ_h, where $\tau_h(o_i) = 1$. After the selection of the first element, $c - 1$ iterations are conducted to select the remaining representatives (Equation 10). Based on $\mathcal{R}$, the initial clusters set $\mathcal{S}$ can be defined, such that $|\mathcal{S}| = c$ and $\forall \mathcal{S}_i \in \mathcal{S},\ \mathcal{S}_i = \{r_i \in \mathcal{R}\}$. Therefore, a unitary cluster is created for each representative object $o_i \in \mathcal{R}$.

3.2.3 Reliable Clusters Set

The hyperedge is a powerful representation of the relationship between multiple data elements. Consequently, this structure can also be explored to represent clusters based on the hyperedges of their respective objects. In this scenario, the *cluster hyperedges* set can be defined as the incidence matrix $\mathbf{H}_s$ of size $|S| \times |V|$, which represents the assignment degree between the data objects and each of the defined clusters. Therefore, $\mathbf{H}_s$ can be obtained as follows:

$$h_s(\mathcal{S}_i, v_j) = \sum_{e_z \in \mathcal{S}_i} h(e_z, v_j), \tag{11}$$

The matrix $\mathbf{H}_s$ supports the agglomeration process that leads to the reliable clusters set. Therefore, its values are updated on each new agglomeration performed in this step. The cluster assignment degree $h_s(\mathcal{S}_i, v_j)$ is given by the sum of similarity values between a data object represented by v_j and all the objects in the cluster $\mathcal{S}_i$. Based on the cluster assignment degree and the size of each cluster, a function $n_c : \mathcal{C} \rightarrow \mathcal{S}$ selects the most similar cluster to an object o_i, being defined as:

$$n_c(o_i) = \underset{\mathcal{S}_j \,\in\, \mathcal{S}}{\arg\max} \; \frac{h_s(\mathcal{S}_j, o_i)}{|\mathcal{S}_j|}. \tag{12}$$

Based on the n_c function, a sub-set of the data collection is agglomerated to retrieve an initial cluster configuration. Let $q = n \times p$ be the size of the sub-set of data collection clustered in this step, where $n = |\mathcal{C}|$ and $p \in (0, 1)$ is a constant. The set containing all objects selected for agglomeration, $\mathcal{C}_a$ is given by the top hyperedge self-confidence score $w_h(\cdot)$, and can be defined as:

$$\begin{aligned} \mathcal{C}_a = \{\mathcal{C}_a \subseteq \mathcal{C} \setminus \mathcal{R}, |\mathcal{C}_a| = q - c \wedge \forall o_i \in \mathcal{C}_a, \\ o_j \in \mathcal{C} \setminus \mathcal{C}_a : w_h(o_i) > w_h(o_j)\}. \end{aligned} \tag{13}$$

By agglomerating the objects contained in $\mathcal{C}_a$, each initial cluster $\mathcal{S}_j \in \mathcal{S}$ can be defined as:

$$\mathcal{S}_j = \bigcup_{o_i \in \mathcal{C}_a \,\bigwedge\, n_c(o_i) = \mathcal{S}_j} \{o_i\} \tag{14}$$

The $n_c(o_i)$ function select the target for an object o_i based on the current cluster configuration. Therefore, the order contained in the ranked list τ_h is exploited to conduct the agglomerations. After the agglomeration of all objects contained in $\mathcal{C}_a$, the SGCC recovers a highly reliable initial cluster configuration, which is used as training labels for a GCN model, responsible to conduct the final clustering.

3.2.4 Graph Convolution Network Clustering

In a final step, the initial reliable cluster set obtained by SGCC is used as soft-labels for training a GCN model. Afterwards, the trained model classify the remaining objects into the discovered clusters. The cluster configuration contained in $\mathcal{S}$ represents the subset of labeled nodes $\mathcal{V}_L$, as defined in Section 2.3. The training procedure uses optimization based on the cross-entropy loss over the labeled nodes set $\mathcal{V}_L$. After being trained, a new inference is executed in order to retrieve the final cluster configuration for the whole input data collection $\mathcal{C}$. Next section discusses GCNs and how they are used by SGCC.

3.3. Graph Convolutional Networks

In recent years, much effort has been made to develop deep learning approaches based on graph data [8]. The GCN main objective can be summarized as to learn an embedding (representation) for each node based on the iterative aggregation of its neighbors, encoding the received graph structure in a neural network model. In a representative work [15], a two-layer GCN model is applied to semi-supervised classification, utilizing a graph represented by a symmetric adjacency matrix $\mathbf{A}$. The obtained model can be defined as a function based both on the feature matrix $\mathbf{X}$ and on the adjacency matrix $\mathbf{A}$:

$$\mathbf{Z} = f(\mathbf{X}, \mathbf{A}), \tag{15}$$

In this scenario, $\mathbf{Z}$ represents an embedding matrix, such that $\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n]^T \in \mathbb{R}^{n \times d}$ and $\mathbf{z}_i$ is a d-dimensional embedded representation computed for the node v_i. In order to obtain the matrix $\mathbf{Z}$, firstly the degree matrices are computed as a pre-processing step, being defined as $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$, where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\tilde{\mathbf{D}}$ is the degree matrix of $\tilde{\mathbf{A}}$. The function $f(\cdot)$, which represents the two-layer GCN model, can be defined as:

$$\mathbf{Z} = log(softmax(\hat{\mathbf{A}}ReLU(\hat{\mathbf{A}}\mathbf{X}\mathbf{W}^{(0)})\mathbf{W}^{(1)})) \tag{16}$$

The neural network weights for an input-to-hidden layer is defined by $\mathbf{W}^{(0)} \in \mathbb{R}^{d \times H}$, where H represents the number of feature maps. Similarly, $\mathbf{W}^{(1)} \in \mathbb{R}^{H \times d}$ is a hidden-to-output weights matrix. Both $\mathbf{W}^{(0)}$ and $\mathbf{W}^{(1)}$ are trained based on the cross-entropy error over the labeled nodes set $\mathcal{V}_L$. After the embedding process, the softmax activation function is applied row-wise in order to obtain the probability distribution over d class labels for each row. By applying the log function on those probabilities, the class with the less negative value in the embedded representation $\mathbf{z}_i$ is assigned as label to the respective node v_i.

3.3.1 GCN Models and Input Data

Mostly grounded by the success of the GCN [15], various related graph convolutional network models have been recently proposed [37, 16, 39, 6, 20, 3, 17]. In this work, besides the original GCN [15], another two approaches were selected based on recent research applications [27] and on results obtained in our initial experiments: the *Simple Graph Convolution (SGC)* [39], a simplified GCN model obtained after the removal of nonlinearities and the collapse of weight matrices between consecutive layers and the *Approximate Personalized Propagation of Neural Predictions (APPNP)* [16], a model that exploits the relationship between GCNs and PageRank by deriving a propagation strategy based on personalized PageRank.

GCN models receive a feature matrix and an adjacency matrix as inputs. In this scenario, our approach uses the input feature matrix $\mathbf{X}$ and adjacency matrix $\mathbf{A}$ is computed as reciprocal k-NN graph based on LHRR's output ranked list set $\mathcal{T}^{(t+1)}$. Natural graph-based data can also be considered for the input graph, as citation network datasets.

4. Experimental Evaluation

4.1. Experimental Protocol

The experimental analysis considered seven different datasets, containing from 3 to 200 classes. The first group of datasets is composed by four image datasets: (*i*) MPEG-7, 1400 images, 70 classes [18]; (*ii*) Flowers, 1360 images, 17 classes [26]; (*iii*) Corel5k, 5000 images, 50 classes [21]; and (*iv*) CUB200, 11788 images, 200 classes [38]. Besides from MPEG-7, which used the distance matrix obtained by CFD [28] descriptor, all image datasets used feature vectors extracted from a Resnet [12] CNN pre-trained on the ImageNet dataset. The input ranked list for the LHRR method was obtained by the Euclidean distance in the image datasets features. The k-NN graph used in the GCNs training was based on the ranked lists obtained from LHRR.

The second group is composed of three citation network datasets, largely used in semi-supervised learning tasks and in recently proposed deep clustering techniques: (*i*) Cora, 2708 articles, 7 classes [34]; (*i*) CiteSeer, 3312 articles, 6 classes [34]; and (*i*) PubMed, 19717 articles, 3 classes [34]. For this group, the original binary feature vectors were used alongside their respective citation graphs. The input ranked list for the LHRR method were obtained by the Jaccard index in the binary feature vectors.

Three external measures are used for effectiveness evaluation: Normalized Mutual Information (NMI) [33], V-Measure [31] and Accuracy (ACC) [24].

4.2. Parameter Settings

Regarding parameter settings, the SGCC method requires only four parameters, c: the number of clusters; k: neighborhood size used by *LHRR* and for the creation of the reciprocal k-NN graph; t: number of iterations for *LHRR*; and p: percentage of data elements agglomerated in the initial reliable clusters. The parameter c followed the size of real classes in all scenarios. The impact of the p parameter was analyzed by varying its value in all the available range while maintaining the remaining configurations. Figure 2 presents the results for NMI and V-Measure metrics on Corel5K dataset, using $k = 50$ and both $t = 1$ and $t = 2$. Both GCN models presented similar behaviors regarding the relationship between parameters p and t.

For $t = 1$, both metrics improved as a larger portion of the dataset was separated before the GCN training. However, using 2 iterations of the LHRR method, all models presented the best results with around 20% to 30% of the dataset being separated as soft labels. Aiming to explore a larger range of configurations, half of the dataset was classified and used as soft-labels in the GCN training ($p = 0.5$) in all experiments.

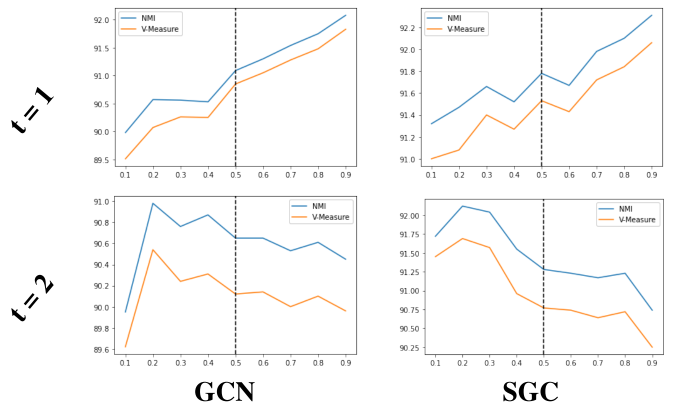

Figure 2. Evaluation of the impact of parameter p. The experiment was conducted on *Corel5K*, using k = 50. The results of *NMI* and *V-Measure* for the GCN models using values t = 1 e t = 2.

Furthermore, two evaluation scenarios were considered: (*i*) using $k = 50$ for every experiment, being this value chosen as an indication with comparable results in all evaluated datasets, and (*ii*) varying k with values between $[10..100]$ and reporting the best results. In both scenarios, the best result for the parameter t between 1 and 2 is reported.

Additionally, during all experiments, the GCN models used 32 hidden layers and learning rate of 10^{-3} being halved every 100 epochs. The model was trained for 400 epochs. The reported values are the mean and standard deviation over 10 executions and every iteration has an early-stop mechanism, in case the model achieves 100% accuracy over the training labels [2].

4.3. Results

The proposed approach was evaluated in two experiments, over seven different datasets. In a first experiment, the neighborhood size was fixed to $k = 50$ in all scenarios. Table 1 presents the results. The SGC model obtained the best value for all metrics in almost every dataset, except for Cora and PubMed, and the GCN model was not able to process the MPEG-7 dataset, probably since the distance matrix was used as a feature matrix.

In a second experiment, the neighborhood size k was varied in the $[10..100]$ range, searching for better results. Table 2 presents the obtained results. Again, in this scenario, the SGC model achieved the best results in almost every dataset [3].

4.4. Comparison with Other Methods

The SGCC was compared with six different clustering approaches, considering traditional and recent methods: K-Means [22], Agglomerative [25], HDBSCAN [23], FINCH [32], SDCN [7] and MinCutPool [5]. FINCH is a recently proposed method based on first-neighbor relations. SDCN and MinCutPool are both based on recent GCN models. All compared methods used the pre-defined parameters and, when possible, the exact number of classes as the desired number of clusters.

[2]The parameter selection is discussed in the supplementary material.

[3]The effect of each component from our proposed approach is analyzed in the ablation study contained in the supplementary material

Table 1. Results for $k = 50$ on all datasets.

Dataset	Network	t	NMI	V-Measure	ACC
	GCN	1	91.34 ± 00.16	91.10 ± 00.15	88.45 ± 00.12
Corel5K	SGC	1	$\mathbf{91.74 \pm 00.06}$	$\mathbf{91.50 \pm 00.06}$	$\mathbf{88.74 \pm 00.05}$
	APPNP	1	91.64 ± 00.15	90.47 ± 00.15	88.72 ± 00.15
	GCN	2	68.93 ± 00.42	68.22 ± 00.45	47.70 ± 00.46
CUB200	SGC	2	$\mathbf{69.97 \pm 00.03}$	$\mathbf{69.33 \pm 00.03}$	$\mathbf{48.37 \pm 00.04}$
	APPNP	2	69.68 ± 00.13	68.93 ± 00.12	47.77 ± 00.30
	GCN	2	80.48 ± 00.57	80.18 ± 00.59	82.61 ± 00.61
Flowers	SGC	2	$\mathbf{81.27 \pm 00.07}$	$\mathbf{81.01 \pm 00.07}$	$\mathbf{83.49 \pm 00.09}$
	APPNP	2	80.79 ± 00.25	80.51 ± 00.25	82.85 ± 00.23
	GCN	-	-	-	-
MPEG-7	SGC	2	$\mathbf{89.74 \pm 00.34}$	$\mathbf{87.64 \pm 00.35}$	$\mathbf{74.06 \pm 00.77}$
	APPNP	1	16.85 ± 33.70	06.83 ± 13.65	02.02 ± 01.19
	GCN	1	31.32 ± 00.40	30.71 ± 00.40	45.86 ± 00.21
Cora	SGC	1	35.94 ± 00.11	35.49 ± 00.11	$\mathbf{49.65 \pm 00.06}$
	APPNP	1	$\mathbf{37.12 \pm 00.23}$	$\mathbf{36.52 \pm 00.24}$	49.18 ± 00.27
	GCN	2	28.80 ± 00.21	28.51 ± 00.21	54.20 ± 00.17
Citeseer	SGC	2	$\mathbf{30.86 \pm 00.07}$	$\mathbf{30.47 \pm 00.07}$	$\mathbf{55.86 \pm 00.05}$
	APPNP	2	30.55 ± 00.17	30.24 ± 00.17	55.68 ± 00.20
	GCN	1	18.32 ± 00.08	$\mathbf{18.04 \pm 00.07}$	$\mathbf{56.45 \pm 00.07}$
PubMed	SGC	1	$\mathbf{31.19 \pm 00.40}$	17.77 ± 00.49	49.76 ± 00.29
	APPNP	1	18.17 ± 00.08	17.69 ± 00.10	56.27 ± 00.38

Table 2. Results considering the best k in a range $[10..100]$.

Dataset	Network	k	t	NMI	V-Measure	ACC
	GCN	95	2	91.89 ± 00.13	91.79 ± 00.13	90.86 ± 00.11
Corel5K	SGC	70	2	$\mathbf{92.62 \pm 00.06}$	$\mathbf{92.44 \pm 00.06}$	90.80 ± 00.04
	APPNP	95	2	92.27 ± 00.12	92.16 ± 00.11	$\mathbf{91.19 \pm 00.12}$
	GCN	55	2	69.07 ± 00.13	68.21 ± 00.12	47.52 ± 00.16
CUB200	SGC	55	2	$\mathbf{69.97 \pm 00.02}$	$\mathbf{69.19 \pm 00.02}$	$\mathbf{48.38 \pm 00.02}$
	APPNP	50	2	69.68 ± 00.13	68.93 ± 00.12	47.77 ± 00.30
	GCN	45	2	80.98 ± 00.28	80.70 ± 00.31	82.68 ± 00.47
Flowers	SGC	50	2	$\mathbf{81.27 \pm 00.07}$	$\mathbf{81.01 \pm 00.07}$	$\mathbf{83.49 \pm 00.09}$
	APPNP	45	2	81.27 ± 00.24	80.99 ± 00.26	82.86 ± 00.33
	GCN	25	1	07.58 ± 22.75	02.32 ± 06.96	01.76 ± 00.99
MPEG-7	SGC	20	2	$\mathbf{96.45 \pm 00.15}$	$\mathbf{96.37 \pm 00.15}$	$\mathbf{94.56 \pm 00.16}$
	APPNP	75	1	32.39 ± 39.81	12.71 ± 20.09	04.82 ± 08.77
	GCN	85	1	39.45 ± 00.34	38.97 ± 00.33	59.22 ± 00.22
Cora	SGC	85	1	$\mathbf{45.02 \pm 00.15}$	$\mathbf{44.81 \pm 00.15}$	$\mathbf{62.96 \pm 00.09}$
	APPNP	65	1	44.58 ± 00.19	44.39 ± 00.18	62.46 ± 00.18
	GCN	60	1	32.84 ± 00.24	32.64 ± 00.24	61.11 ± 00.20
Citeseer	SGC	65	1	$\mathbf{35.50 \pm 00.08}$	$\mathbf{35.34 \pm 00.08}$	$\mathbf{62.95 \pm 00.07}$
	APPNP	65	1	35.42 ± 00.21	35.23 ± 00.21	62.73 ± 00.14
	GCN	40	1	26.31 ± 00.07	$\mathbf{25.37 \pm 00.07}$	$\mathbf{62.95 \pm 00.11}$
PubMed	SGC	45	1	$\mathbf{28.59 \pm 00.08}$	22.41 ± 00.07	52.15 ± 00.04
	APPNP	65	1	27.41 ± 00.39	25.12 ± 00.13	62.07 ± 00.41

Table 3 presents the results obtained on image datasets. All the best results inside the same standard deviation are highlighted. During the evaluation process, we were not able to obtain effective results for MPEG-7 with the SDCN method and for CUB200 with the MinCutPool method. It can be observed that the proposed SGCC achieved the best results for all metrics in all datasets. Table 4 presents the results on the citation network datasets. Analogously, all the best results inside the same standard deviation are highlighted. In this scenario, the proposed SGCC approach achieved the best results on Cora and CiteSeer datasets for all measures. On PubMed dataset, SGCC achieved the best results considering the ACC measure.

4.5. Visual Analysis

In the visual analysis, we employed dimensionality reduction methods to represent the impact of the proposed method on a 2-D projection of feature space, considering the GCN-based embeddings. Figure 3 presents the visual-

Table 3. Comparison of NMI, V-Measure (VM) and ACC values in image datasets between *SGCC*, classic and recent clustering methods.

Method	Input	Corel5K			CUB200			Flowers			MPEG-7		
		NMI	VM	ACC	NMI	VM	ACC	NMI	VM	ACC	NMI	VM	ACC
K-Means (1967)	X	89.38 ±00.46	88.71 ±00.42	82.12 ±01.07	67.07 ±00.19	66.45 ±00.14	41.03 ±00.44	73.23 ±00.93	72.90 ±00.99	71.85 ±02.06	79.32 ±00.32	78.65 ±00.37	59.55 ±00.93
Agglomerative (1983)	X	91.03	90.65	86.68	67.06	66.24	42.03	78.06	77.03	72.64	90.43	86.76	59.00
HDBSCAN (2017)	X	75.66	54.91	35.28	49.94	14.89	04.30	38.60	15.98	13.52	90.16	79.32	64.92
FINCH (2019)	X	90.06	81.13	52.32	77.23	25.65	04.57	79.60	66.54	52.20	87.04	83.72	60.64
SDCN (2020)	X & A	87.43 ±00.36	86.95 ±00.32	81.51 ±00.74	62.62 ±00.21	61.23 ±00.18	31.76 ±00.62	67.02 ±00.99	66.73 ±00.99	36.91 ±00.58	-	-	-
MinCutPool (2020)	X & A	85.76 ±00.78	77.71 ±17.28	33.96 ±12.38	-	-	-	72.55 ±02.05	72.46 ±02.07	74.54 ±02.82	30.07 ±36.89	06.59 ±08.62	00.02 ±00.69
Proposed Approach													
SGCC (GCN)	X & A	91.89 ±00.13	91.79 ±00.13	90.86 ±00.11	69.07 ±00.13	68.21 ±00.12	47.52 ±00.16	80.98 ±00.28	**80.70 ±00.31**	82.68 ±00.47	07.58 ±22.75	02.32 ±06.96	01.76 ±00.99
SGCC (SGC)	X & A	**92.62 ±00.06**	**92.44 ±00.06**	90.80 ±00.04	**69.97 ±00.02**	**69.19 ±00.02**	**48.38 ±00.02**	**81.27 ±00.07**	**81.01 ±00.07**	**83.49 ±00.09**	**96.45 ±00.15**	**96.37 ±00.15**	**94.56 ±00.16**
SGCC (APPNP)	X & A	92.27 ±00.12	92.16 ±00.11	**91.19 ±00.12**	69.68 ±00.13	68.93 ±00.12	47.77 ±00.30	**81.27 ±00.24**	**80.99 ±00.26**	82.86 ±00.33	32.39 ±39.81	12.71 ±20.09	04.82 ±08.77

Table 4. Comparison of NMI, V-Measure (VM) and ACC results in citation datasets between *SGCC*, classic and recent clustering methods.

Method	Input	Cora			Citeseer			PubMed		
		NMI	VM	ACC	NMI	VM	ACC	NMI	VM	ACC
K-Means (1967)	X	16.80 ±04.80	15.60 ±04.57	35.50 ±03.08	18.65 ±04.79	18.19 ±04.81	40.57 ±06.16	**35.46 ±00.07**	**31.26 ±00.08**	59.51 ±00.01
Agglomerative (1983)	X	23.39	21.93	37.22	19.76	18.97	42.23	11.75	04.04	42.59
HDBSCAN (2017)	X	04.84	00.39	29.87	**40.01**	01.29	21.52	01.38	00.06	39.84
FINCH (2019)	X	20.38	01.84	30.46	26.78	15.06	32.94	05.39	01.64	40.38
SDCN (2020)	X & A	21.65 ±00.16	21.17 ±00.16	38.49 ±00.18	30.96 ±00.10	30.69 ±00.10	58.09 ±00.10	07.64 ±00.28	00.02 ±00.00	39.94 ±00.00
MinCutPool (2020)	X & A	41.68 ±01.96	40.41 ±01.90	39.43 ±01.82	28.51 ±02.78	28.20 ±02.75	35.01 ±02.34	20.66 ±20.29	20.29 ±01.10	46.84 ±02.76
Proposed Approach										
SGCC (GCN)	X & A	39.45 ±00.34	38.97 ±00.33	59.22 ±00.22	32.84 ±00.24	32.64 ±00.24	61.11 ±00.20	26.31 ±00.07	25.37 ±00.07	**62.95 ±00.11**
SGCC (SGC)	X & A	**45.02 ±00.15**	**44.81 ±00.15**	**62.96 ±00.09**	**35.50 ±00.08**	**35.34 ±00.08**	**62.95 ±00.07**	28.59 ±00.08	22.41 ±00.07	52.15 ±00.04
SGCC (APPNP)	X & A	44.58 ±00.19	44.39 ±00.19	62.46 ±00.18	**35.42 ±00.21**	**35.23 ±00.21**	62.73 ±00.14	27.41 ±00.39	25.12 ±00.13	62.07 ±00.41

ization of the original features and the GCN-based features from two datasets: Flowers and Cora. The *t-SNE* [36] algorithm was used for dimensionality reduction.

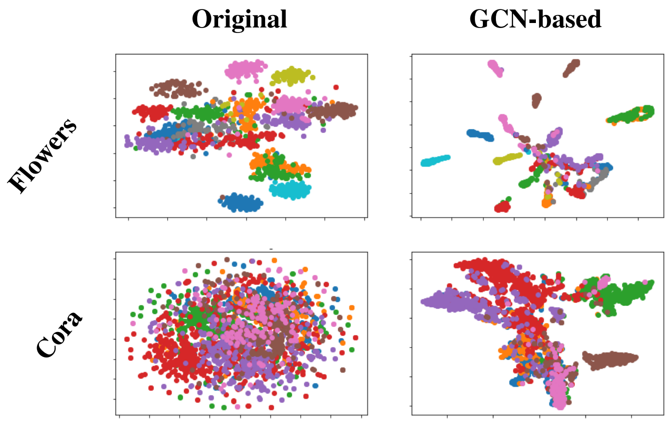

Figure 3. Visual analysis: the impact of the GCN embeddings.

It is possible to notice that the GCN-based embeddings improved the separability among classes, setting the elements in the same class closer in the two-dimension embedding space. The positive effects can be observed in both datasets, especially in the Cora dataset where the original features, represented as binary vectors, are not able to encode the class similarity information. Therefore, the separation is highly improved by the GCN-based embeddings.

5. Conclusion

In this work, a novel self-supervised clustering approach is proposed. The method combines a hypergraph-based manifold learning method, able for modeling similarity information in unsupervised scenarios, with GCN models, which represent the state-of-the-art in semi-supervised classification. The SGCC method was evaluated both in image and citation networks datasets and compared with classic and recent clustering methods. The results obtained were better in most of the scenarios. Additionally, visual analyses were conducted to illustrate the embedding efficiency of our proposed approach. As future work, we intend to further analyze the parameter estimation aiming to pre-select the best values and to further explore the hypergraph information to enhance the obtained results.

Acknowledgements

The authors are grateful to São Paulo Research Foundation - FAPESP (grants #2017/25908-6 and #2018/15597-6), Brazilian National Council for Scientific and Technological Development - CNPq (grants #309439/2020-5 and #422667/2021-8), Petrobras (grant #2017/00285-6) and QuintoAndar for financial support.

References

[1] Luis Claudio Sugi Afonso, Daniel Carlos Guimarães Pedronette, André N. de Souza, and João Paulo Papa. Improving optimum- path forest classification using unsupervised manifold learning. In *24th International Conference on Pattern Recognition, ICPR*, pages 560–565, 2018.

[2] Filipe Alves de Fernando., Daniel Pedronette., Gustavo José de Sousa., Lucas Valem., and Ivan Guilherme. Rade: A rank-based graph embedding approach. In *Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP,*, pages 142–152. INSTICC, SciTePress, 2020.

[3] Song Bai, Feihu Zhang, and Philip H. S. Torr. Hypergraph convolution and hypergraph attention. *CoRR*, abs/1901.08150, 2019.

[4] Abla Chouni Benabdellah, Asmaa Benghabrit, and Imane Bouhaddou. A survey of clustering algorithms for an industrial context. *Procedia Computer Science*, 148:291–302, 2019.

[5] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Spectral clustering with graph neural networks for graph pooling. In *Proceedings of the 37th international conference on Machine Learning (ICML)*, pages 2729–2738. ACM, 2020.

[6] Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. Graph neural networks with convolutional ARMA filters. *CoRR*, abs/1901.01343, 2019.

[7] Deyu Bo, Xiao Wang, Chuan Shi, Meiqi Zhu, Emiao Lu, and Peng Cui. Structural deep clustering network. In *Proceedings of The Web Conference (WWW) 2020*, 2020.

[8] HongYun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Trans. Knowl. Data Eng.*, 30(9):1616–1637, 2018.

[9] Olivier Chapelle, Bernhard Schlkopf, and Alexander Zien. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010.

[10] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E. Hinton. Big self-supervised models are strong semi-supervised learners. *CoRR*, abs/2006.10029, 2020.

[11] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. *CoRR*, abs/2011.10566, 2020.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[13] Zijian Hu, Zhengyu Yang, Xuefeng Hu, and Ram Nevatia. Simple: Similar pseudo label exploitation for semi-supervised classification. *CoRR*, abs/2103.16725, 2021.

[14] Anil K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, jun 2010.

[15] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[16] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations, ICLR 2019*, 2019.

[17] Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. In *Advances in Neural Information Processing Systems, NeurIPS 2019*, pages 13333–13345, 2019.

[18] Longin Jan Latecki, Rolf Lakamper, and Ulrich Eckhardt. Shape descriptors for non-rigid shapes with a single closed contour. In *CVPR*, pages 424–429, 2000.

[19] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[20] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18)*, pages 3538–3545. AAAI Press, 2018.

[21] Guang-Hai Liu and Jing-Yu Yang. Content-based image retrieval using color difference histogram. *Pattern Recognition*, 46(1):188 – 198, 2013.

[22] James B. MacQueen. Some methods for classification and analysis of multivariate observations. 1967.

[23] Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, 2, 03 2017.

[24] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, and J. Long. A survey of clustering with deep learning: From the perspective of network architecture. *IEEE Access*, 6:39501–39514, 2018.

[25] Fionn Murtagh. A survey of recent advances in hierarchical clustering algorithms. *The Computer Journal*, 26(4):354–359, 1983.

[26] M-E. Nilsback and A. Zisserman. A visual vocabulary for flower classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1447–1454, 2006.

[27] Daniel Pedronette and Longin Jan Latecki. Rank-based self-training for graph convolutional networks. *Information Processing Management*, 58:102443, 03 2021.

[28] Daniel Carlos Guimarães Pedronette and Ricardo da Silva Torres. Shape retrieval using contour features and distance optimization. In *VISAPP (2)*, pages 197–202. Citeseer, 2010.

[29] Daniel Carlos Guimarães Pedronette, Lucas Pascotti Valem, Jurandy Almeida, and Ricardo da S. Torres. Multimedia retrieval through unsupervised hypergraph-based manifold ranking. *IEEE Transactions on Image Processing*, 28(12):5824–5838, 2019.

[30] Daniel Carlos Guimarães Pedronette, Lucas Pascotti Valem, and Ricardo da S. Torres. A bfs-tree of ranking references for unsupervised manifold learning. *Pattern Recognition*, 111:107666, 2021.

[31] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 410–420, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[32] Saquib Sarfraz, Vivek Sharma, and Rainer Stiefelhagen. Efficient parameter-free clustering using first neighbor relations. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[33] Amit Saxena, Mukesh Prasad, Akshansh Gupta, Neha Bharill, Om Prakash Patel, Aruna Tiwari, Meng Joo Er, Weiping Ding, and Chin-Teng Lin. A review of clustering techniques and developments. *Neurocomputing*, 267:664 – 681, 2017.

[34] Prithviraj Sen, Galileo Mark Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective

classification in network data. *AI Magazine*, 29(3):93–106, 2008.

[35] Zixing Song, Xiangli Yang, Zenglin Xu, and Irwin King. Graph-based semi-supervised learning: A comprehensive review. *arXiv preprint arXiv:2102.13303*, 2021.

[36] Laurens van der Maaten and Geoffrey Hinton. Viualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 11 2008.

[37] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[38] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[39] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International Conference on Machine Learning (ICML)*, volume 97, pages 6861–6871, 2019.

[40] Yaochen Xie, Zhao Xu, Jingtun Zhang, Zhengyang Wang, and Shuiwang Ji. Self-supervised learning of graph neural networks: A unified review. *arXiv preprint arXiv:2102.10757*, 2021.