Journal of Algorithms and Computation

journal homepage: http://jac.ut.ac.ir

Overlapping Clusters in Cluster Graph Convolutional Networks

Mahmood Amintoosi*[1]

[1]Faculty of Mathematics and Computer science, Hakim Sabzevari University, Sabzevar, IRAN.

ABSTRACT

A popular research topic in Graph Convolutional Networks (GCNs) is to speedup the training time of the network. The main bottleneck in training GCN is the exponentially growing of computations. In Cluster-GCN based on this fact that each node and its neighbors are usually grouped in the same cluster, considers the clustering structure of the graph, and expand each node's neighborhood within each cluster when training GCN. The main assumption of Cluster-GCN is the weak relation between clusters; which is not correct at all graphs. Here we extend their approach by overlapped clustering, instead of crisp clustering which is used in Cluster-GCN. This is achieved by allowing the marginal nodes to contribute to training in more than one cluster. The evaluation of the proposed method is investigated through the experiments on several benchmark datasets. The experimental results show that the proposed method is more efficient than Cluster-GCN, in average.

ARTICLE INFO

Article history:
Research paper
Received 01, September 2021
Received in revised form 12, November 2021
Accepted 28 November 2021
Available online 30, December 2021

Keyword: Networks, Graph Neural Networks, Clustering, Spectral Clustering.

AMS subject Classification: 91C20, 68T07, 92B20, 05C90.

*Corresponding author: M. Amintoosi. Email: m.amintoosi@hsu.ac.ir

1 Introduction

Convolutional Neural Networks (CNNs) has been successfully applied to many computer vision applications [1, 2, 13]. Convolutions and pooling operator are the main tools of these networks for extracting local features and using a pyramid structure from the input signal. Graph Convolutional Networks (GCNs) [15] are similar tools for acting on graphs as input signals. GCNs extend the convolution operation from regular domains to arbitrary topologies and unordered structures [4]. As in CNNs, graph pooling is an important operation that allows a GCNs to learn representations of the input graphs, by summarizing local components. Recently GCNs has been widely used in many graph-based applications, including community detection [19], semi-supervised classification [15] and Recommender Systems [24]. GCNs like CNNs have many layers; in which at each layer, the embedding of a node is obtained by gathering the embeddings of its neighbors, followed by one or few layers of linear transformations and activations. With this graph convolution operation, GCN is able to learn useful node representations via Stochastic Gradient Descent (SGD).

Some emerging research for GCN is focusing on speeding up the training of GCN. One way to speed up the training is to use mini-batch. While one of the computation issue for mini-batch is due to the neighborhood explosion over layers. GCN training speedup algorithms are thus mainly focusing on how to perform neighbors sampling so that to avoid heavy neighborhood searching within each batch. For example, FastGCN[6] samples neighboring nodes within each layer. GraphSAGE [12] considers sampling only a fixed-size neighborhood samples for each node in the mini-batch, so only a fixed size of neighborhoods are expanded for each batch. Stochastic GCN with variance reduction(VRGCN)[9] reuses the embedding from the previous epoch; however this could increase the memory usage during training because it needs to store temporary embedding for each layer for each node. Cluster-GCN [8] considers the clustering structure of the graph, and expand each node's neighborhood within each cluster when training GCN. This can significantly improve the training computation, since the heavy neighborhood searching outside cluster is dropped. Among many types of introduced GCNs, in this paper we focus on Cluster-GCN [8]; In Cluster-GCN training is done on clusters of the graph, which can make better time and memory complexity, with respect to traditional methods; in which the gathering information process takes place independently, at the same time for all the nodes. Here, the main idea is to use overlapped clusters, in contrast to [8] that uses non-overlapping clusters. The experimental results showed the efficiency of the proposed method. Before explaining the proposed method, we first briefly review the theoretical background of GCNs and the proposed based method Cluster-GCN.

2 Background

Since the paper is related to GCNs, a special type of GCN, named Cluster-GCN and overlapping clustering, here we review these subjects briefly.

2.1 Graph Convolutional Network

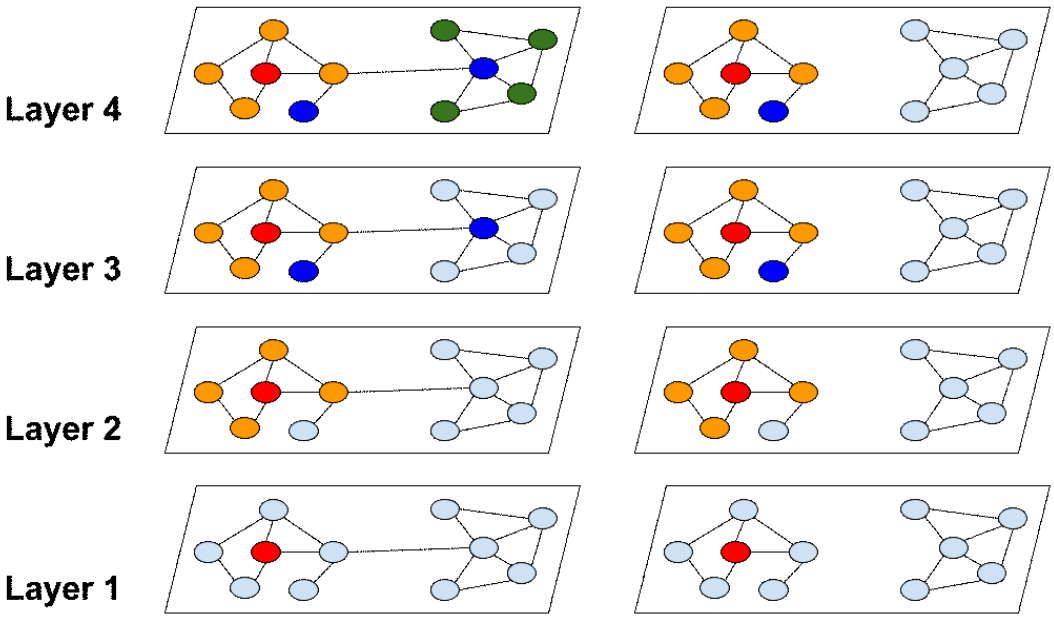
Consider a multi-layer Graph Convolutional Network with the following layer-wise propagation rule [15]:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) . \tag{2.1}$$

Where, $\tilde{A} = A + I_N$ is the adjacency matrix of the undirected graph $\mathcal{G}$ with added self-connections. I_N is the identity matrix, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ and $W^{(l)}$ is a layer-specific trainable weight matrix. $\sigma(\cdot)$ denotes an activation function, such as the $\mathrm{ReLU}(\cdot) = \max(0, \cdot)$. $H^{(l)} \in \mathbb{R}^{N\times D}$ is the matrix of activations in the l^{th} layer; $H^{(0)} = X$. It has been showed that the form of this propagation rule can be motivated via a first-order approximation of localized spectral filters on graphs [15, 5].
Despite the superior performance on various tasks, training GCN remains computational and memory intensive. The main bottleneck in training GCN is the exponentially growed computations when applying the Stochastic Gradient Descent (SGD) training algorithm. Due to the graph convolution operator, a node embedding at the L^{th} layer depends on the $(L-1)^{th}$ layer embeddings of all its neighbors, which again depends on the $(L-2)^{th}$ layer embeddings of all its neighbors' neighbors and so on. This leads to exponential grown computations even when only one or few loss terms are sampled in SGD. For learning an L layer GCN with a degree-d graph, the computational complexity for updating the loss function of one node is $O(d^L)$, which blows up easily for large graphs or for deeper GCNs. Several approaches have been proposed recently to resolve this exponential growing issue and to speed up GCN training. [12] proposed to speed up graph convolution operation by only sampling a subset of neighbors for each node, also known as the Neighborhood Sampling (NS) approach. However, NS requires a relative large sample size to maintain the performance of GCN. In [7] a variance reduction technique is proposed that can sample only few neighbors for each node, while approximating the embeddings of other nodes using the previously stored embeddings. However, all of these previous attempts still require some neighborhood samples and still have complexity exponential to number of layers. In GraphSAGE [12], to allow the mini-batch training, the authors also provide a variant by uniformly sampling a fixed size of the neighboring nodes for each node. In Cluster-GCN [8] an efficient algorithm for training GCNs is proposed by exploiting the graph clustering structure, which explained in the next sub-section.

2.2 Cluster-GCN

In Cluster-GCN [8] to exploit the local structure in graph, they sample one or few clusters at each iteration and restrict the neighborhood within selected clusters to conduct SGD updates. This strategy leads to huge computational benefits. Figure 1 (from [8]) shows the main idea of Cluster-GCN. As can be seen the Cluster-GCN can avoid heavy neighborhood search by focusing on the neighbors within each cluster. In this figure, the clusters have weak dependencies, and ignoring the neighborhoods of some marginal nodes may not affect the overall embeddings of these nodes. For example in the top left image of figure 1, an orange node, has four neighbors including two blue neighbors. Since one blue node lied



(a) Traditional GCN (b) Cluster-GCN

Figure 1: Figure 1 of [8]. The neighborhood expansion difference between traditional graph convolution and the cluster approach introduced in [8]. Here it is supposed that the clusters have weak connection.

in another cluster, cluster-GCN stopped expansion from this node, hence only this node does not contribute to embedding of the aforementioned orange node; but *what happens if the clusters have more connections and the number of such blue neighbors, belonging to other clusters were not negligible?* The core idea of this paper is investigation of this situation.
In the proposed method we use an NMF base overlapping clustering method, the next sub-section is devoted to it.

2.3 Overlapping Clustering

Overlapping clustering is well known in community analysis domain. Communities are densely connected subnetworks; the task of community detection is to find the community structure of a given network. Many real-world networks exhibit overlapping community structure in which vertices may belong to more than one community[18].
Non-negative Matrix Factorization (NMF) has been widely adopted for community detection due to its great interpret-ability and its natural fitness for capturing the community membership of nodes. The equivalency of Nonnegative Matrix Factorization and K-means has been shown in [10, 14]. In K-means clustering, the objective function to be minimized is the sum of squared distances from each data point to its centroid. With $A = [a_1, \ldots, a_n] \in \mathbb{R}^{m \times n}$, the objective function J_k with given integer k can be written as:

$$J_k = \sum_{j=1}^{k} \sum_{a_i \in C_j} \|a_i - c_j\|^2 \tag{2.2}$$

Where $a_i \in \mathbb{R}^m$ is a data point (or a node of graph) and $c_j \in \mathbb{R}^m$ is the center of j^{th} cluster C_j. The above objective function can be written as follows:

$$\sum_{i} \left\|a_i - c_{\sigma_j}\right\|^2 = \|A - CH\|_F^2 \tag{2.3}$$

where $\sigma_i = j$ when i^{th} point is assigned to j^{th} cluster ($j \in \{1, \ldots, k\}$), $C^{m \times k}$ is the clusters' centers and the column j of $H^{k \times n}$, demonstrate the membership degrees of data point a_i to each of k clusters. Each data point a_i is represented by a linear combination of cluster centers. Regrading eq. (2.3), objective functions for K-means and NMF may look the same; however, constraints are different:

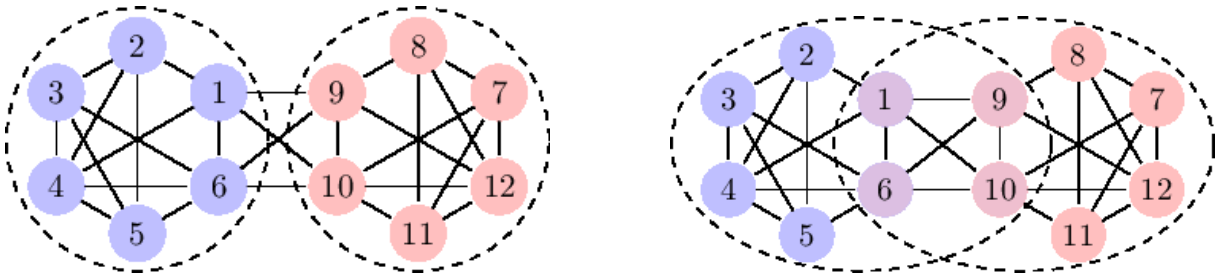
- K-means: $H \in \{0, 1\}^{k \times n}, \mathbf{1}_k^T H = \mathbf{1}_n^T$
- NMF: $W \geq 0, H \geq 0$

As can be seen, NMF may be considered as a soft clustering, in contrast to K-means, which is a hard clustering method. NMF is the basis of many overlapping community detection methods. In community detection, if node $u \in V$ represents a data point, each entry $H_{c,u}$ represents the weight between node $u \in V$ and community c. The larger $H_{c,u}$

is, the more possible that u belongs to c. On the other hand, if $H_{c,u}$ is 0, u does not belong to c.
Some of the community detection methods that can detect overlapped and non-overlapped communities are as follows: In [21] the community assignment has been done by mapping the original network to the community membership space by NMF. Modularized Nonnegative Matrix Factorization (M-NMF) model [22] preserves both the microscopic structure (pairwise node similarity) and community structure for network embedding. The procedure uses joint non-negative matrix factorization with modularity based regularization in order to learn a cluster membership distribution over nodes. The method can be used in an overlapping and non-overlapping way. In [23] a method named DANMF (Deep Autoencoder-like Nonnegative Matrix Factorization) for community detection is proposed The procedure uses non-negative matrix factorization in order to learn a cluster membership distribution over nodes. DANMF consists of an encoder component and a decoder component. This architecture empowers DANMF to learn the hierarchical mappings between the original network and the final community assignment. [11] uses biased second order random walks to approximate the point-wise mutual information matrix obtained by pooling normalized adjacency matrix powers. This matrix is decomposed by an approximate factorization technique.
Regarding the good performance of the DANMF, we will use this method in the next section as a tool for overlapping clustering. Note that each clustering method that support overlapping may be used instead of it.

3 Overlapped Cluster-GCN

We saw that in Cluster-GCN a crisp clustering is performed and whether between the partitions have weak or strong connections, the neighborhood expansion will be stopped at clusters' borders. Here we allow to extend the neighborhood expansion a bit more to near clusters by using Cluster-GCN with overlapped clusters, instead of crisp clustering. Consider the crisp clustering showed in figure 2(a); the crisp clustering produces two non-overlapping clusters. GCN training is done in every cluster, separately. Hence, although nodes 9 and 10 are neighbors of node 1, they will not affecting the embedding of node 1, resulting that node 1, losses 2 of 5 neighbors' influences. As can be seen, the nodes 1 and 6, that belong to the left cluster, may be also considered as members of the right cluster, with some degree of membership score. Many clustering algorithms use hard (crisp) partitioning techniques where each object is assigned to one cluster; but some algorithms utilize overlapping techniques where an object may belong to one or more clusters [3]. Figure 2(b), shows an overlapped clustering of the mentioned graph. Here Nodes 1 and 6 at the same time are belong to two clusters, hence these nodes and nodes 9 and 10 are contributed to train procedure of two clusters. With this trick, with preserving the benefit of clustering technique of Cluster-GCN, the neighborhood expansion will be more efficient for border nodes.
Based on the H, described in section 2.3, we can extract the community membership



(a) Non-overlapped clustering (b) Overlapped clustering

Figure 2: A graph with 12 nodes, and two non-overlapped and overlapped clusters. In contrast to figure 1 which is supposed that clusters have weak connection, some times as this example, the clusters may be strong connections. Nodes 1, 6, 9 and 10 are overlapping nodes.

of nodes. For disjoint community detection, each node is assigned to the community where it gets the largest belonging propensity. For overlapping community detection, we need to set a threshold δ in order to determine whether a node belongs to a community or not. If $H_{c,u} \geq \delta$, we say that node u belongs to community c. Suppose that *winner* indicates the cluster number that node u has the greatest membership to it, i.e. $winner = \arg\max_c H_{c,u},\ c \in \{1, \ldots, k\}$. For a crisp (non-overlapping) clustering, these winner index, for each node, indicates its cluster number; for overlapping clusters, based on a threshold δ the node u may belongs to some other clusters. Here we set this threshold as a related value to $H_{winner,u}$. We defined the coefficient WMC (Winner Membership Closeness) as the percentage that another cluster membership value, is closed to the winner cluster. With this coefficient, node u is belongs to all clusters c, that $H_{c,u} \geq WMC \times H_{winner,u}$. If $WMC = 0$, node u is belongs to all clusters, and if $WMC = 1$, u belongs only to the *winner* class, which is the crisp clustering.
Here the DANMF[23] method is used as the overlapping communities detection approach. Although DANMF was the best method among 12 community detection compared in [23], but note that here the goal is neither selecting the best non-overlapping clustering method nor the best number of clusters or some other related parameters. The main idea is the improvement Cluster-GCN by allowing to clusters to overlap with each other which causes better embeddings expansion. With this idea the Cluster-GCN algorithm is modified and showed in algorithm 1. The experimental results in the next section verified this idea. Although theoretical discussion of Cluster-GCN algorithm in [8] is based on the non-overlapped clusters, but it is not necessary for implementation; training can be done in every cluster, separately.

4 Experimental Results

In [8] the superior performance of Cluster-GCN against some STOA methods including FastGCN [6], GraphSAGE [12], VR-GCN [7], GaAN [25], GAT [20] and GeniePath

Algorithm 1: Overlapped Cluster-GCN

Input: Graph A, feature X, label Y;
Output: Node representation $\bar{X}$, Learned Weights W.
Partition graph nodes into c **Overlapped** clusters $\mathcal{V}_1, \mathcal{V}_2, \cdots, \mathcal{V}_c$ by DANMF [23];
for $iter = 1, \cdots,$ *max_iter* **do**
  Randomly choose q clusters, $t_1, \cdots, t_q$ from $\mathcal{V}$ without replacement;
  Form the subgraph $\bar{G}$ with nodes $\bar{\mathcal{V}} = [\mathcal{V}_{t_1}, \mathcal{V}_{t_2}, \cdots, \mathcal{V}_{t_q}]$ and links $A_{\bar{\mathcal{V}},\bar{\mathcal{V}}}$;
  Compute $g \leftarrow \nabla \mathcal{L}_{A_{\bar{\mathcal{V}},\bar{\mathcal{V}}}}$ (loss on the subgraph $A_{\bar{\mathcal{V}},\bar{\mathcal{V}}}$) ;
  Conduct Adam update using gradient estimator g
Output: $\{W_l\}_{l=1}^{L}$

[16] is shown, hence we compare our proposed approach only with the Cluster-GCN method. Some the citation network datasets are used here which are accessible from `PyTorch-geometric` site[1]. Table 1 shows the specifications of the used datasets.
The implementation here[2] is with `PyTorch`[3] and is based on `KarateClub`[4] library [17] and a PyTorch implementation of Cluster-GCN which is accessible from Github[5] repository. The parameters of GCN in all experiments are as follows:

Number of Layers: 3,

Shape of Layers: [Number of features, 16, 16 ,16 , Number of labels]

Optimization Method: Adam Optimizer

Clustering Method: DANMF [23], with number of pre-iterations equal 500

Number of Clusters: 2×Number of labels

Number of Epochs: or *max_iter* in Algorithm 1 is 10

Number of random clusters in each run: It is equal to number of total clusters ($q = c$ in Algorithm 1)

Train-Test split: 30%, 70%

Dropout: 50%

In [15], the authors conducted some experiments with shallow and deep GCNs on first 3 datasets of tables 1. From the figure 5 of [15], we see that the best results are obtained with a 2- or 3-layer model. Hence here for all datasets a 3 layer model is utilized. All

[1] https://pytorch-geometric.readthedocs.io/en/latest/modules/datasets.html
[2] https://github.com/mamintoosi/Overlapping-Cluster-GCN
[3] https://pytorch.org/
[4] https://karateclub.readthedocs.io/
[5] https://github.com/benedekrozemberczki/ClusterGCN

Table 1: Data statistics

Datasets	#Nodes	#Edges	#Labels	#Features
CiteSeer	3,327	9,104	6	3,703
Cora	2,708	10,556	7	1,433
PubMed	19,717	88,648	3	500
WikiCS	11,701	297,110	10	300

Table 2: Non-overlapped (crisp) clustering [8] versus overlapped clustering (The proposed method). The proposed method leads to better performance (in terms of test F1 score), $WMC = 0.2$.

Dataset	Crisp Clustering [8]	Overlapping Clustering
CiteSeer	0.68	**0.7**
Cora	0.80	**0.82**
Pubmed	0.53	**0.82**
WikiCS	0.51	**0.64**

the experiments are executed on Google Colab machine with a NVIDIA Tesla K80 GPU (11.5 GB memory), Intel Xeon CPU (2.20 GHz), and 13.3 GB of RAM.
As mentioned in section 3, WMC (Winner Membership Closeness), is defined as a coefficient that indicates whether other clusters memberships of this node are close enough to the winner cluster or not. If $WMC = 0$, the node is belongs to all cluster and if $WMC = 1$, the node is only belongs to the winner cluster (i.e. non-overlapping clustering). In the following experiments, we run the proposed method on $WMC = 0.1, 0.2, \ldots, 0.9, 1$. $WMC = 1$ corresponds to Cluster-GCN approach[8] which uses crisp clustering.
DANMF[23], in which has the best performance on first 3 datasets in table 1 is used here for community detection. This algorithm can be used for overlapping and non-overlapping community detection. Any other community detection approach also may be used here instead of DANMF, which is not the main concern of this paper.
F1 score, which is the common metric uses in the field is selected for quantitative comparison. The results on test nodes with $WMC = 0.2$ are summarized in Table 2. As can be seen the proposed overlapping Cluster-GCN achieved the best accuracy for these datasets.
Figure 3 shows the F1 score, when WMC varies from 0.1 to 1. As can be seen, in average, increasing WMC, reduces F1-score, which means that less overlapping, gains lower score. Note that in all situations, more overlapping, does not equal to higher F1-score; for example in 3(a), for `PubMed` dataset, overlapped cluster in $WMC = 0.8$ produced lower F1 score than $WMC = 0.9$; hence the efficiency of the proposed method is related to the data and the overlapping rate. For an instance graph data that varies over time, a good overlapping rate can be found once and used many times.
Figure 4(a) shows the rate of overlapping nodes in various rate of overlapping. $WMC = 1$ represents crisp clustering i.e. zero overlapping. In non-overlapping clustering ($WMC =$

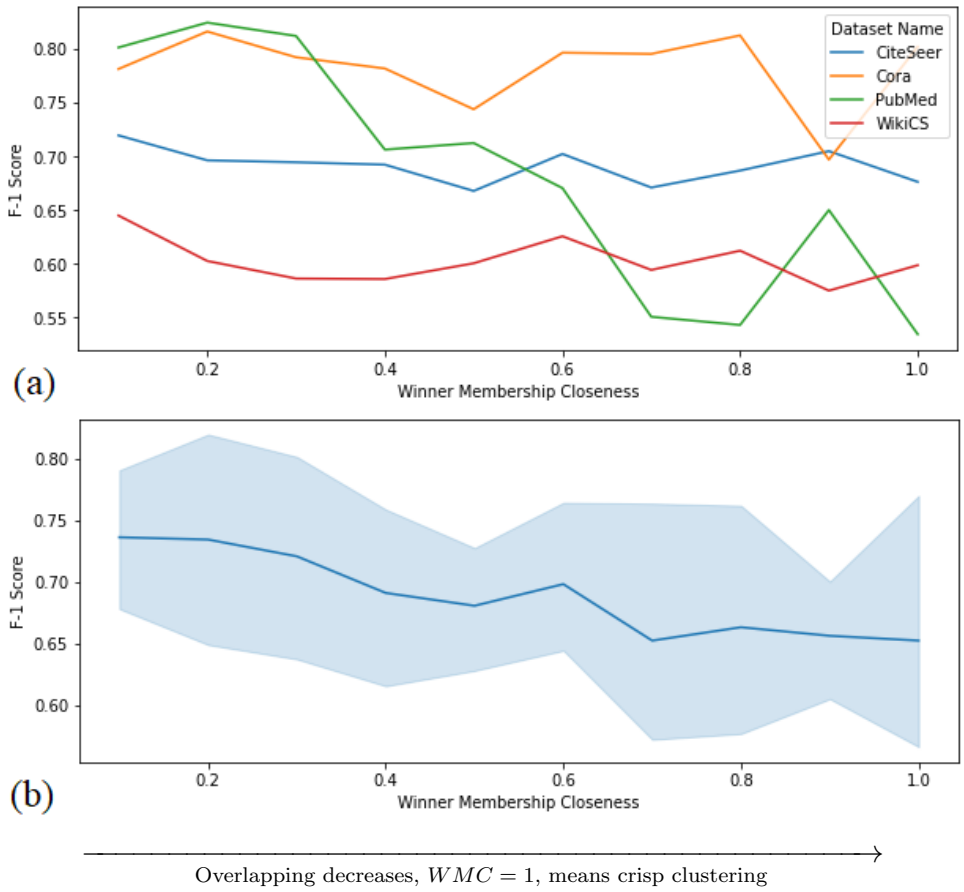
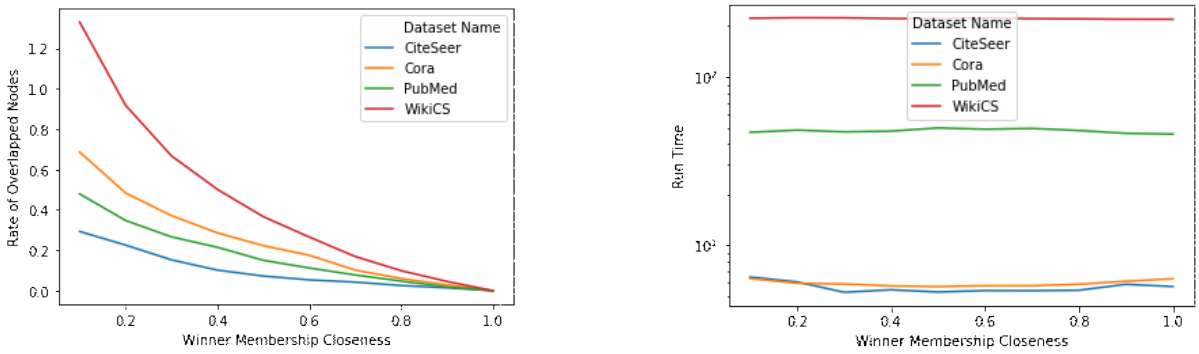


Figure 3: The effect of overlapping clusters in F1-score, related to Winner Membership Closeness (WMC). WMC less than 1, show the proposed approach. WMC equal 1, is corresponds to non-overlapping clusters. In average, increasing WMC, reduced F1-score. Sub-figure (a), shows the results for each dataset, separately, (b), shows the average.



(a) Percentage of overlapped (b) Run Times

Figure 4: Rate of overlapped nodes and run times.

1) the total number of clusters' nodes is equal to the number of graph nodes, hence this rate is equal zero. As expected, lower WMC, yields higher rate. In `PubMed` dataset and $WMC = 0.1$, this rate is equal 0.5, which means, in average about 50% of nodes, were appeared in more than one cluster. But this increasing the total number of nodes contributing in training procedure, did not affected the run time. Figure 4(b) shows the running times. As can be seen, run times on various WMC's are almost identical in each dataset; note that $WMC = 1$ is crisp clustering.

5 Conclusion

In this paper a modified version of cluster-GCN [8] was proposed. In contrast to [8] that the clusters were non overlapped, here it is allowed to clusters to overlapped to each other, resulting that the neighborhood expansion may be extended beyond the crisp non overlapped clusters. The experimental results showed the superior efficiency of the proposed approach. Although in the proposed approach –as like as the base method and many other GCNs – various parameters and sub modules such as number of intermediate layers, activation functions, clustering methods and so on may be investigated, but these subjects are not lie in the main scope of this paper; the main claim of this paper is that this modification of Cluster-GCN, may improve it, which experimental results verified it. The superior performance of Cluster-GCN against some STOA approach were shown previously in [8], and hence the performance of the proposed method is compared only with the Cluster-GCN.

References

[1] Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., and Farhan, L. Review of deep learn-

ing: concepts, cnn architectures, challenges, applications, future directions. Journal of Big Data, 8(1), (2021) 53.

[2] Amintoosi, M. Combining a regularization method and the optimal brain damage method for reducing a deep learning model size. Journal of Machine Vision and Image Processing, 9(1), (2022) 31–45.

[3] Baadel, S., Thabtah, F., and Lu, J. Overlapping clustering: A review. In 2016 SAI Computing Conference (SAI), (2016) 233–237.

[4] Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. Relational inductive biases, deep learning, and graph networks. arXiv preprint arXiv:1806.01261 (2018).

[5] Bianchi, F. M., Grattarola, D., and Alippi, C. Spectral clustering with graph neural networks for graph pooling. In Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event, vol. 119 of Proceedings of Machine Learning Research, (2020). 874–883.

[6] Chen, J., Ma, T., and Xiao, C. FastGCN: Fast learning with graph convolutional networks via importance sampling. In ICLR (2018a).

[7] Chen, J., Zhu, J., and Le, S. Stochastic training of graph convolutional networks with variance reduction. In ICML (2018b).

[8] Chiang, W.-L., Liu, X., Si, S., Li, Y., Bengio, S., and Hsieh, C.-J. Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,(2019) 257–266.

[9] Dai, H., Kozareva, Z., Dai, B., Smola, A., and Song, L. Learning steady-states of iterative algorithms over graphs. In ICML, (2018) 1114–1122.

[10] Ding, C., He, X., and Simon, H. D. On the Equivalence of Nonnegative Matrix Factorization and K-means – Spectral Clustering (2005) 606–610.

[11] Grover, A., and Leskovec, J. Node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, New York, NY, USA: Association for Computing Machinery. (2016) 855–864.

[12] Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. In Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, Red Hook, NY, USA: Curran Associates Inc. (2017) 1025–1035.

[13] He, K., Gkioxari, G., Dollár, P., and Girshick, R. Mask R-CNN. IEEE Transactions on Pattern Analysis and Machine Intelligence, 42(2), (2020) 386–397.

[14] Kim, J., and Park, H. Sparse nonnegative matrix factorization for clustering. Technical report, College of Computing, Georgia Institute of Technology, USA. (2008).

[15] Kipf, T. N., and Welling, M. Semi-supervised classification with graph convolutional networks. In ICLR. (2017).

[16] Liu, Z., Chen, C., Li, L., Zhou, J., Li, X., Song, L., and Qi, Y. Geniepath: Graph neural networks with adaptive receptive paths. In AAAI. (2019).

[17] Rozemberczki, B., Kiss, O., and Sarkar, R. Karate Club: An API Oriented Open-source Python Framework for Unsupervised Learning on Graphs. In Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), ACM. (2020) 3125–3132.

[18] Shang, J., Liu, L., Xie, F., and Wu, C. How overlapping community structure affects epidemic spreading in complex networks. In 2014 IEEE 38th International Computer Software and Applications Conference Workshops, (2014) 240–245.

[19] Sun, J., Zheng, W., Zhang, Q., and Xu, Z. Graph neural network encoding for community detection in attribute networks. IEEE Transactions on Cybernetics, (2021) 1–14.

[20] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. (2018).

[21] Wang, F., Li, T., Wang, X., Zhu, S., and Ding, C. Community discovery using nonnegative matrix factorization. Data Mining and Knowledge Discovery, 22(3), (2011) 493–521.

[22] Wang, X., Cui, P., Wang, J., Pei, J., Zhu, W., and Yang, S. Community preserving network embedding. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, AAAI'17, (2017) 203–209.

[23] Ye, F., Chen, C., and Zheng, Z. Deep autoencoder-like nonnegative matrix factorization for community detection. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, New York, NY, USA: Association for Computing Machinery. (2018) 1393–1402.

[24] Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., and Leskovec, J. Graph convolutional neural networks for web-scale recommender systems. In KDD (2018)..

[25] Zhang, J., Shi, X., Xie, J., Ma, H., King, I., and Yeung, D.-Y. GaAN: Gated attention networks for learning on large and spatiotemporal graphs. In UAI (2018).