arXiv:2006.16904v1 [cs.LG] 30 Jun 2020

Graph Clustering with Graph Neural Networks

Anton Tsitsulin[*] University of Bonn
John Palowitch Google Research
Bryan Perozzi Google Research
Emmanuel Müller University of Bonn

Abstract

Graph Neural Networks (GNNs) have achieved state-of-the-art results on many graph analysis tasks such as node classification and link prediction. However, important *unsupervised* problems on graphs, such as graph clustering, have proved more resistant to advances in GNNs. In this paper, we study *unsupervised* training of GNN pooling in terms of their clustering capabilities.

We start by drawing a connection between graph clustering and graph pooling: intuitively, a good graph clustering is what one would expect from a GNN pooling layer. Counterintuitively, we show that this is not true for state-of-the-art pooling methods, such as MinCut pooling. To address these deficiencies, we introduce Deep Modularity Networks (DMoN), an unsupervised pooling method inspired by the modularity measure of clustering quality, and show how it tackles recovery of the challenging clustering structure of real-world graphs. In order to clarify the regimes where existing methods fail, we carefully design a set of experiments on synthetic data which show that DMoN is able to jointly leverage the signal from the graph structure and node attributes. Similarly, on real-world data, we show that DMoN produces high quality clusters which correlate strongly with ground truth labels, achieving state-of-the-art results.

1 Introduction

In recent years there has been a surge of research interest in developing varieties of Graph Neural Networks (GNNs) – specialized deep learning architectures for dealing with graph-structured data, such as social networks [47], recommender graphs [68], or molecular graphs [13, 69]. GNNs leverage the structure of the data as computational graph, allowing the information to propagate across the edges of graphs [49]. When many real-wold systems are represented as graphs, they exhibit locally inhomogeneous distributions of edges, forming *clusters* (also called *communities* or *modules*) – groups of nodes with high in-group edge density, and relatively low out-group density. Clusters can correspond to interesting phenomena in the underlying graph, for example to education [57] or employment [41] in social graphs. GNNs have been shown to benefit from leveraging higher-order structural information that could arise from clusters [9, 69, 31], for example through pooling or trainable attention over edges [60].

Interestingly, most existing work on GNNs to leverage higher-order structure does not directly address node partitioning or the estimation of clusters within the computational graph. Furthermore, most works explore these mechanisms only within a semi-supervised or supervised framework, ignoring the fact that *unsupervised* graph clustering is often an extremely useful end-goal in itself – whether for data exploration [45], visualization [11, 12], genomic feature discovery [7], anomaly detection [44], or for many other use-cases discussed e.g. in [19]. Additionally, many of the existing structure-aware methods have undesirable properties, such as relying on a multi-step optimization process which does not allow to optimize the objective via gradient descent end-to-end [46].

[*]Work done while interning at Google.

Preprint. Under review.

Table 1: Related work in terms of six desirable clustering properties outlined in Section 2.

method	Trainable	Unsupervised	Sparse	Node pooling	Soft assignments	Stable
Graclus [13]	✘	✔	✔	✔	✘	✔
DiffPool [69]	✔	✔	✘	✔	✔	✘
Top-k [20]	✔	✘	✘	✘	✘	✔
SAG [31]	✔	✘	✘	✘	✘	✔
MinCut [4]	✔	✔	✔	✔	✔	✘
DMoN	✔	✔	✔	✔	✔	✔

In this work, we take an *ab initio* approach to the clustering problem in the GNN domain, bridging the gap between traditional graph clustering objectives and deep neural networks. We start by drawing a connection between graph pooling, which was typically studied in the literature as a regularizer for supervised GNN architectures, and fully unsupervised clustering. Specifically, we contribute:

- DMoN, a method for unsupervised clustering in GNNs that allows optimization of cluster assignments in an end-to-end differentiable way.
- An empirical study of performance on synthetic graphs, illustrating the problems with existing work and how DMoN allows for improved model performance in those regimes.
- Thorough experimental evaluation on real-world data, showing that many pooling methods poorly reflect hierarchical structures and are not able to make use of both graph structure and node attributes.

2 Related Work

Our work builds upon a rich line of research on graph neural networks and graph pooling methods.

Graph Neural Networks (GNNs) [49, 15, 40, 21, 29] allow end-to-end differentable losses over data with arbitrary structure. They have been applied to an incredible range of applications, from social networks [47], to recommender systems [68], to computational chemistry [21]. While GNNs are flexible enough to allow for unsupervised losses, most work follows the semi-supervised setting for node classification from [29]. For a complete introductions to the vast topic we refer interested readers to detailed surveys [6, 25, 8].

Unsupervised training of GNNs is commonly done via maximizing mutual information [3, 26, 58, 54] in a self-supervised fashion. Deep Graph Infomax (DGI) [61] adapted the mutual informaton-based learning from Deep InfoMax [26], learning unsupervised representations for nodes in attributed graphs. InfoGraph [56] extended the idea to learning representations of whole graphs instead of just nodes. A very similar approach was introduced independently in [27] in the context of pre-training GNNs for producing representations of graphs, which was first tackled in [36].

Graph pooling aims to tackle the hierarchical nature of graphs via iterative coarsening. Early architectures [13] resorted to fixed axiomatic pooling, with no optimization of clustering while the network learns. DiffPool [69] suggests to include a *learnable* pooling to GNN architecture. To help the convergence, DiffPool includes a link prediction loss to help encapsulate the clustering structure of graphs and an additional entropy loss to penalize soft assignments. Top-k [20] and SAG pooling [31] learn to sparsify the graph (select top-k edges for each node) with learned weights. MinCutPool [4] pooling studies differentiable formulation of spectral clustering as a pooling strategy.

We summarize mainstream graph pooling methods in Table 1 in terms of six desirable properties related to their clustering capabilities:

- **Trainable.** End-to-end training allows to capture both graph structure and node features.
- **Unsupervised** training is a desirable setting for clustering models. Works on *supervised* graph clustering [66, 62] are outside of the scope of this work.
- **Sparse.** As graphs in the real-world vary in size and sparsity, methods can not be limited by a $\mathcal{O}(n^2)$ link prediction objectives, like DiffPool [69], or computing $\mathbf{A}^2$, like top-k pooling methods [20, 31] .
- **Node aggregation** is crucial for our interpretation of graph pooling in terms of graph clustering. Both Top-k [20] and SAG pooling [31] only sparsify the graph and do not reduce the nodeset.
- **Soft assignments** allow for more flexibility reasoning about the interactions of clusters.
- **Stable** – the method should be stable in terms of the graph structure. DiffPool [69] is not stable in terms of graph sparsity, while MinCutPool [4] can not deal with uneven degree distribution.

Graph embeddings [48, 23, 59] can be thought of as (very restricted) unsupervised GNNs with an identity feature matrix, meaning each node learns its own positional representation [70]. The learning process in graph embeddings is often done in a similar way to DGI though noise contrastive estimation [24]. As far as we know, all pooling strategies for learning node embeddings without attributes have been axiomatic [10, 33, 14].

3 Preliminaries

We introduce the necessary background for DMoN, starting with the problem formulation, reviewing common graph clustering objectives and how they can be made differentiable efficiently.

Graph $G = (V, E)$ is defined via a set of nodes $V = (v_1, \ldots, v_n), |V| = n$ and edges $E \subseteq V \times V, |E| = m$. We are interested in measuring the quality of graph partitioning function $\mathcal{F} : n \to k$ that splits the set of nodes V into k partitions $V_i = \{v_j, \mathcal{F}(v_j) = i\}$. Additionally, in contrast to standard graph clustering, we are also provided with node attributes $\mathbf{X} \in \mathbb{R}^{n \times s}$ that provide additional information not reflected in the graph structure, but also correlated with it.

3.1 Graph Clustering Quality Functions

Design of the objective function is crucial for the algorithm performance. We review two families of clustering quality functions amenable to spectral optimization, and review some of their shortcomings.

Cut-based metrics. In his seminal work [18], Fiedler suggested that the second (Fiedler) eigenvector of a graph Laplacian produces a graph *cut* minimal in terms of the weight of the edges. This plain notion of cut degenerates on real-world graphs, as it does not require partitions to be balanced in terms of size. It is possible to get normalized partitions with the use of ratio cut [63], which normalizes the cut by the product of the number of nodes in two partitions, or normalized cut [52], which uses total edge volume of the partition as normalization.

In real networks, however, there is evidence *against* existence of good cuts [32] in ground-truth communities. This can be explained by the fact than a single node implicitly participates in many different clusters [17], e.g. a person in a social network is simultaneously connected with family and work friends, forcing the algorithm to merge these communities together.

Recently, MinCutPool [4] adapted the notion of the normalized cut to use as a regularizer for pooling. While, theoretically, MinCutPool objective should be suitable for clustering nodes in graphs, we (i) show that it does not optimize its own objective function and (ii) provide evidence against this in the synthetic and real-world experiments.

Modularity [38] approaches the same problem from a statistical perspective, incorporating a *null model* to quantify the significance of the clustering with respect to the random graph. In a fully random graph with given degrees, nodes u and v with degrees d_u and d_v are connected with probability $d_u d_v / 2m$. Modularity measures the divergence between the intra-cluster edges from the expected one:

$$\mathcal{Q} = \frac{1}{2m} \sum_{ij} \left[\mathbf{A}_{ij} - \frac{d_i d_j}{2m} \right] \delta(c_i, c_j), \tag{1}$$

where $\delta(c_i, c_j) = 1$ if i and j are in the same cluster and 0 otherwise. Note $\mathcal{Q} \in (-1/2; 1]$ (it is 0 when there is no correlation of clusters with edge density), but it is not necessarily maximized at 1, and is only comparable across graphs with the same degree distribution. While problems with the modularity metric have been identified [22], it remains one of the most commonly-used and eminently useful graph clustering metrics in scientific literature [19].

3.2 Spectral Modularity Maximization

Maximizing the modularity is proven to be **NP**-hard [5], however, a spectral relaxation of the problem can be solved efficiently [37]. Let $\mathbf{C} \in 0, 1^{n \times k}$ be the cluster assignment matrix and $\mathbf{d}$ be the degree vector. Then, with *modularity matrix* $\mathbf{B}$ defined as $\mathbf{B} = \mathbf{A} - \frac{\mathbf{d}\mathbf{d}^\top}{2m}$, the modularity $\mathcal{Q}$ can be reformulated as:

$$\mathcal{Q} = \frac{1}{2m} \operatorname{Tr}(\mathbf{C}^\top \mathbf{B} \mathbf{C}) \tag{2}$$

Relaxing $\mathbf{C} \in \mathbb{R}^{n \times k}$, the optimal $\mathbf{C}$ maximizing $\mathcal{Q}$ is the top-k eigenvectors of the modularity matrix $\mathbf{B}$. While $\mathbf{B}$ is dense, iterative eigenvalue solvers can take advantage of the fact that $\mathbf{B}$ is a sum of a sparse $\mathbf{A}$ and rank-one matrix $-\frac{\mathbf{d}\mathbf{d}^\top}{2m}$, meaning that the matrix-vector product $\mathbf{B}\mathbf{x}$ can be computed efficiently as

$$\mathbf{B}\mathbf{x} = \mathbf{A}\mathbf{x} - \frac{\mathbf{d}^\top \mathbf{x} \mathbf{d}}{2m}$$

and optimized efficiently with iterative methods such as power iteration or Lanczos algorithm. One can then obtain clusters by means of spectral bisection [37] with iterative refinement akin to Kernighan-Lin algorithm [28]. However, these formulations operate entirely on the graph structure, and it is non-trivial to adapt them to work with attributed graphs.

3.3 Graph Neural Networks

Graph Neural Networks are a flexible class of models that perform nonlinear feature aggregation with respect to graph structure. For the purposes of this work, we consider transductive GNNs that output a single embedding per node. Graph convolutional networks (GCNs) [29] are simple yet effective [51] message-passing networks that fit our criteria. Let $\mathbf{X}^0 \in \mathbb{R}^{n \times s}$ be the initial node features and $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ be the normalized adjacency matrix, the output of t-th layer $\mathbf{X}^{t+1}$ is

$$\mathbf{X}^{t+1} = \mathrm{SeLU}(\tilde{\mathbf{A}}\mathbf{X}^t\mathbf{W} + \mathbf{X}\mathbf{W}_{\mathrm{skip}}) \tag{3}$$

We make two changes to the classic GCN architecture: first, we remove the self-loop creation and instead use an $\mathbf{W}_{\mathrm{skip}} \in \mathbb{R}^{s \times s}$ trainable skip connection, and, second, we replace ReLU nonlinearity with SeLU [30] for better convergence.

4 Method

In this section, we present DMoN, our method for attributed graph clustering with graph neural networks. Inspired by the modularity function and its spectral optimization, we propose a fully differentiable unsupervised clustering objective which optimizes soft cluster assignments using a null model to control for inhomogeneities in the graph. We then discuss the challenge of regularizing cluster assignments, and present collapse regularization, a softer version of orthogonality loss, that against trivial solutions.

4.1 DMoN: Deep Modularity Networks

The challenge of clustering boils down to defining an optimization procedure over the cluster assignment matrix $\mathbf{C}$. In DMoN, we propose to obtain $\mathbf{C}$ via the output of a softmax function, which allows the (soft) cluster assignment to be differentiable. The input to the cluster assignment can be any differentiable message passing function, but here we specifically consider the case where a graph convolutional network is used to obtain soft clusters for each node as follows:

$$\mathbf{C} = \mathrm{softmax}(\mathrm{GCN}(\tilde{\mathbf{A}}, \mathbf{X})), \tag{4}$$

where GCN is a (possibly) multi-layer convolutional network operating on an normalized adjacency matrix $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{A})\mathbf{D}^{-\frac{1}{2}}$.

We then propose to optimize this assignment with the following objective, which combines insights from spectral modularity maximization (2) with a regularization to ensure informative clusters:

$$\mathcal{L}_{\mathrm{DMoN}} = \underbrace{-\frac{1}{2m}\mathrm{Tr}(\mathbf{C}^\top \mathbf{B}\mathbf{C})}_{\text{modularity}} + \underbrace{\frac{\sqrt{k}}{n}\left\|\sum_i \mathbf{C}_i^\top\right\|_F - 1}_{\text{collapse regularization}}, \tag{5}$$

where $\|\cdot\|_F$ is the Frobenius norm. We decompose the computation of $\mathrm{Tr}(\mathbf{C}^\top\mathbf{B}\mathbf{C})$ as a sum of sparse matrix-matrix multiplication and rank-one degree normalization $\mathrm{Tr}(\mathbf{C}^\top\mathbf{A}\mathbf{C} - \mathbf{C}^\top\mathbf{d}^\top\mathbf{d}\mathbf{C})$. This allows us to efficiently optimize DMoN parameters in sparse regime.

4.2 Collapse regularization

Without additional constraints on the assignment matrix $\mathbf{C}$, spectral clustering for both min-cut and modularity objectives has spurious local minima: assigning all nodes to the same cluster produces a trivial locally optimal solution that traps gradient-based optimization methods. MinCut pooling [4] addresses this problem by adapting spectral orthogonality constraint in the form of soft-orthogonality [2] regularization $\left\|\mathbf{C}^\top\mathbf{C} - \mathbf{I}\right\|_F$. We notice that this term is overly restrictive when combined with softmax class assignment – intuitively, when the value range of $\mathbf{C}$ is restricted to $\mathbb{R} \cap [0, 1]$ the optimization of the soft-orthogonality regularizer dominates the loss.

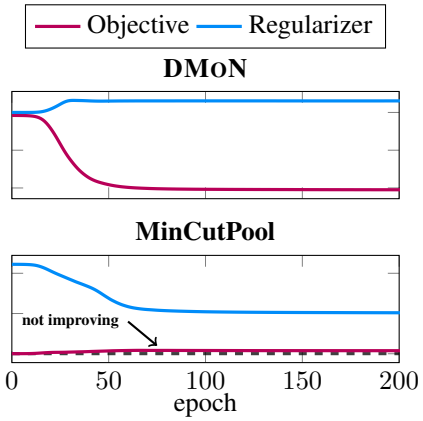


Figure 1: Optimization progress of MinCut and DMoN on Cora dataset. MinCut optimizes the regularizer, while DMoN minimizes its main objective.

We illustrate this problem in Figure 1, which depicts the progress of optimization for both methods in terms of their main objective and regularizer term over the course of 200 epochs on Cora dataset. The soft orthogonality regularization term dominates the optimization for MinCutPool, such that the cut objective becomes *worse than random* over the course of training.

We fix this problem by proposing a relaxed notion of *collapse regularization* that prevents the trivial partition while not restricting the optimization of the main objective. The regularizer is a Frobenius norm of the (soft) cluster membership counts, normalized to range $[0, 1]$. It gets value of 0 when cluster sizes are perfectly balanced, and 1 in the case all clusters collapse to one. We also propose to stabilize the training by applying dropout [55] to GNN representations before the softmax, preventing the gradient descent from getting stuck in the local optima of the highly non-convex objective function.

5 Experiments

We assess the clustering performance of DMoN in terms of both graph clustering and label alignment. We release the implementation of DMoN at this URL[2].

Figure 2: Dataset statistics.

dataset	Nodes	Edges	Features	Classes
Cora	2708	5278	1433	7
Citeseer	3327	4614	3703	6
Pubmed	19717	44325	500	3
Amazon PC	13752	143604	767	10
Amazon Photo	7650	71831	745	8
Coauthor CS	18333	81894	6805	15
Coauthor PHY	34493	247962	8415	5

Datasets. We use 7 real-world datasets for assessing model quality. Cora, Citeseer, and Pubmed [50] are citation networks; nodes represent papers connected by citation edges; features are bag-of-word abstracts, and labels represent paper topics. Amazon PC and Amazon Photo [51] are subsets of the Amazon co-purchase graph for the computers and photo sections of the website, where nodes represent goods with edges between ones frequently purchased together; node features are bag-of-word reviews, and class labels are product category. Coauthor CS and Coauthor PHY [51] are co-authorship networks based on the Microsoft Academic Graph for the computer science and physics fields respectively; nodes are authors, which are connected by edge if they co-authored a paper together; node features are a collection of paper keywords for author's papers; class labels indicate most common fields of study.

Baselines. As we study how to leverage the information from both the graph and attributes, we employ two baselines that employ either strictly graph or attribute information. We give a brief description of the baselines used below:

- **k-means(features)** is our baseline that only considers the feature data. We use the local Lloyd algorithm [34] with the k-means++ seeding [1] strategy.
- **SBM** [42] is a baseline that only relies on the graph structure. We estimate a constrained stochastic block model with given number of k clusters, maximizing the modularity [38] of the network.
- **k-means(DGI)** [61] demonstrates the need of joint learning of clusters and representations. We learn unsupervised node representation with DGI and run k-means on the resulting representations.
- **MinCutPool(graph, features)** [4] is a deep pooling method that we re-interpret as clustering.

[2] `github.com/google-research/google-research/tree/master/graph_embedding/dmon`

Table 2: Results on four datasets from [50] in terms of graph conductance $\mathcal{C}$, modularity $\mathcal{Q}$, NMI with ground-truth labels, and pairwise F1 measure. We highlight best neural method performance.

	Cora				**Citeseer**				**Pubmed**			
	graph		*labels*		*graph*		*labels*		*graph*		*labels*	
method	$\mathcal{C}\downarrow$	$\mathcal{Q}\uparrow$	NMI↑	F1↑	$\mathcal{C}\downarrow$	$\mathcal{Q}\uparrow$	NMI↑	F1↑	$\mathcal{C}\downarrow$	$\mathcal{Q}\uparrow$	NMI↑	F1↑
k-means(features)	61.7	19.8	18.5	27.0	60.5	30.3	24.5	29.2	55.8	33.4	19.4	24.4
SBM	15.4	77.3	36.2	30.2	14.2	78.1	15.3	19.1	39.0	53.5	16.4	16.7
k-means(DGI)	28.0	64.0	**52.7**	40.1	17.5	73.7	**40.4**	39.4	82.9	9.6	22.0	26.4
MinCut	23.3	70.3	35.8	25.0	14.1	78.9	25.9	20.1	29.6	63.1	25.4	15.8
DMoN	**12.2**	**76.5**	48.8	**48.8**	**5.1**	**79.3**	33.7	**43.2**	**17.7**	**65.4**	**29.8**	**33.9**

Table 3: Results on four datasets from [51] in terms of graph conductance $\mathcal{C}$, modularity $\mathcal{Q}$, NMI with ground-truth labels, and pairwise F1 measure. We highlight best neural method performance.

	Amazon PC				**Amazon Photo**				**Coauthor CS**				**Coauthor PHY**			
	graph		*labels*		*graph*		*labels*		*graph*		*labels*		*graph*		*labels*	
method	$\mathcal{C}\downarrow$	$\mathcal{Q}\uparrow$	NMI↑	F1↑	$\mathcal{C}\downarrow$	$\mathcal{Q}\uparrow$	NMI↑	F1↑	$\mathcal{C}\downarrow$	$\mathcal{Q}\uparrow$	NMI↑	F1↑	$\mathcal{C}\downarrow$	$\mathcal{Q}\uparrow$	NMI↑	F1↑
k-m(feat)	84.5	5.4	21.1	19.2	79.6	10.5	28.8	19.5	49.1	23.1	35.7	39.4	57.0	19.4	30.6	42.9
SBM	31.0	60.8	48.4	34.6	18.6	72.7	59.3	47.4	20.3	72.7	58.0	47.7	25.9	66.9	45.4	30.4
k-m(DGI)	61.9	22.8	22.6	15.0	51.5	35.1	33.4	23.6	35.1	57.8	64.6	51.9	38.6	51.2	51.0	30.6
MinCut	—	—	—	—	—	—	—	—	22.7	70.5	64.6	47.8	27.8	64.3	48.3	24.9
DMoN	**18.0**	**59.0**	**49.3**	**45.4**	**12.7**	**70.1**	**63.3**	**61.0**	**17.5**	**72.4**	**69.1**	**59.8**	**18.8**	**65.8**	**51.9**	**37.0**

Metrics. We measure both the graph-based metrics of clustering and label correlation to study clustering performance of attributed graphs both in terms of graph and attribute structure. For graph-level metrics, we report average cluster conductance (as per definition from [64]) and graph modularity [38]. For ground-truth label correlation analysis, we report normalized mutual information (NMI) between the cluster assignments and labels and pairwise F-1 score between all node pairs and their associated cluster pairs. Where possible, we normalize all metrics by multiplying them by 100 for ease of comparison.

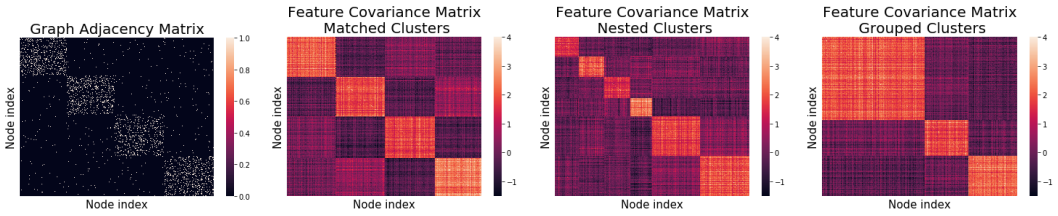
Parameter settings. We run all experiments for 10 times and average results across runs. All models were implemented in Tensorflow 2 and trained on CPUs. We fix the architecture for all clustering networks to have one hidden layer with 512 neurons in it for real-world datasets and 64 for small synthetic graphs. We set the number of clusters to 16 for all datasets and methods.

5.1 Simulation Experiments on Stochastic Block Model

To better explore the robustness of our approach to variance in the graph and node features, we propose a study on synthetic graphs using an *attributed*, *degree-corrected* stochastic block model (ADC-SBM). The SBM [53] plants a partition of clusters ("blocks") in a graph, and generates edges via a distribution conditional on that partition. This model has been used extensively to benchmark graph clustering methods [19], and has recently been used for experiments on state-of-the-art GNNs [16]. In our version of the model, node features are also generated, using a multivariate mixture model, with the mixture memberships having some correspondence to the cluster memberships.

Table 4: Synthetic ADC-SBM benchmark scenarios.

Scenario	Parameter	Description
1	$d_{out} \in [2.0, 5.0]$	Increase graph cluster mixing signal. Higher = weaker clusters.
2	$\sigma_c \in [10^{-2}, 10^{1}]$	Increase feature cluster center variance. Higher = stronger clusters.
3	$\sigma_c \in [10^{-2}, 10^{1}]$	Increase feature cluster center variance, with nested feature clusters.
4	$\sigma_c \in [10^{-2}, 10^{1}]$	Increase feature cluster center variance, with grouped feature clusters.
5	$d \in [2^2, 2^7]$	Increase average degree. Higher = clearer graph signal.
6	$d_{max} \in [2^2, 2^{10}]$	Increase power law upper-bound. Higher = more extreme power law.



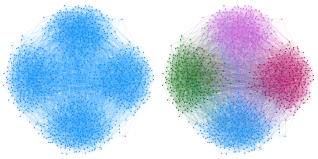
(a) (b) (c) (d)

Figure 3: **Illustration of synthetic data.** a): 4-cluster graph adjacency matrix. b): Covariance matrix of "matched" features: features that are clustered according to the graph clusters. c): Covariance matrix of "nested" features: features that are clustered by nesting of the graph clusters. d): Covariance matrix of "grouped" features: features that are clustered by grouping of the graph clusters.

To generate an instance of the ADC-SBM, we fix a number of nodes n and a number of clusters k, and choose node cluster memberships uniformly-at-random. Define the matrix $\mathbf{D}_{k\times k}$ where $\mathbf{D}_{ij}$ is the expected number of edges shared between nodes in clusters i and j. We determine $\mathbf{D}$ by fixing (1) the expected *average* degree of the nodes $d \in \{1, n\}$, and (2) the expected *average* sub-degree $d_{out} \leq d$ of a node to any cluster other than its own. Note that the difference $d_{in} - d_{out}$, where $d_{in} := d - d_{out}$, controls the spectral detectability of the clusters [35]. Finally, we generate a power-law n-vector θ, where θ_i is proportional to i's expected degree. With the memberships and the parameters D and θ. We use `graph-tool` [43] generate the graphs.

To generate s-dimensional features, we first generate feature memberships from k_f cluster labels. For graph clustering GNNs that operate both on edges and node features, it is important to examine performance on data where feature clusters diverge from or segment the graph clusters: thus potentially $k_f \neq k$. We examine cases where feature memberships *match*, *group*, or *nest* the graph memberships, as illustrated in Figure 3. With feature memberships in-hand, we generate k zero-mean feature cluster centers from a s-multivariate normal with covariance matrix $\sigma_c^2 \cdot \mathbf{I}_{s\times s}$. Then, for feature cluster $i \leq k_f$, we generate its features from a s-multivariate normal with covariance matrix $\sigma^2 \cdot \mathbf{I}_{s\times s}$. Note that the ratio σ_c^2/σ^2 controls the expected value of the classical between/within-sum-of-squares of the clusters.

The above paragraphs describe a single generation of our synthetic benchmark model, the ADC-SBM. To explore robustness, we define a "default" ADC-SBM, and explore model parameters in a range around the defaults. We configure our default model as follows: we generate graphs with $n = 1,000$ nodes grouped in $k = 4$ clusters, and $s = 32$-dimensional features grouped in $k_f = 4$ matching feature clusters with $\sigma = 1$ intra-cluster center variance and $\sigma_c = 3$ cluster center variance. We try to model real-world graphs' degree distribution with $d = 20$ average degree and $d_{out} = 2$ average inter-cluster degree with power law parameters $d_{min} = 2, d_{max} = 4, \alpha = 2$. In total, we consider 6 different scenarios, as described in Table 4.



(a) (b)

Figure 4: (a) MinCutPool fails to recover ground-truth clusters while (b) DMoN perfectly recovers ground-truth partition.

Figure 5 demonstrates overwhelming superiority of DMoN in all considered scenarios. In scenario 1 we see that DMoN can effectively leverage the feature signal to obtain outstanding clustering performance even when the graph structure is close to random, far beyond the spectral detectability threshold (pictured in gray). Scenario 2 demonstrates that even in the presence of a weak feature signal DMoN outperforms stochastic SBM minimization, while MinCutPool needs *two orders of magnitude* stronger signal to attain the same performance. Scenario 3 demonstrates that DMoN is not susceptible to nested feature clusters and can correctly recover underlying graph structure. Scenario 4 shows that DMoN is more susceptible to grouped features, as it becomes hard to differentiate graph clusters with similar features, however, MinCutPool is struggling to pick up *any* signal in the data. Scenarios 5&6 demonstrate that DMoN is stable in terms of the degree distribution of the graph, while MinCutPool is not. We also notice that while the k-means(DGI) baseline offers some improvements over using features or the graph structure alone, it never surpasses the strongest signal provider in the graph, never being better than the best one between k-means(features) and SBM. Finally, we provide an illustration of the how MinCutPool collapsing into a single cluster while DMoN succeeds in Figure 4.

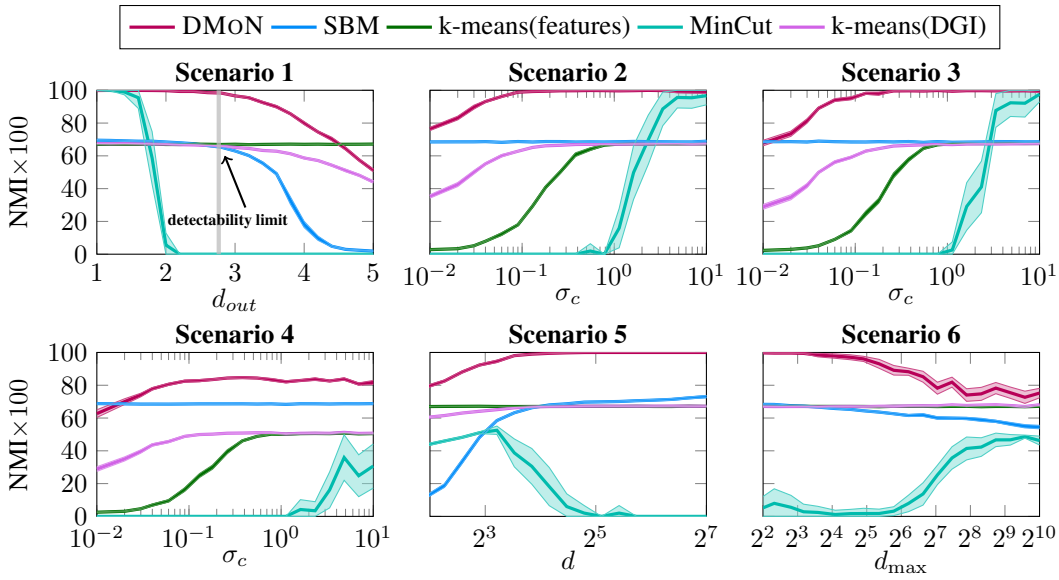


Figure 5: Synthetic results on the ADC-SBM model with 6 different scenarios described in Table 4.

5.2 Real-world Data

We now move on to studies on real-world networks, featuring DMoN and 4 baselines on 7 different datasets. DMoN achieves better clustering performance than its neural counterparts on every single dataset and metric besides losing twice to DGI+k-means on Cora and Citeseer in terms of NMI. Compared to SBM, a method that exclusively optimizes modularity, we are able to stay within 2 NMI percentage points, while simultaneously clustering the features. Surprisingly, on Citeseer and Pubmed we achieve better modularity than the method designed to optimize it – we attribute that to very high correlation between graph structure and the features. We also highlight that we beat MinCutPool in terms of conductance (average graph cut) on all datasets, even though it attempts to optimize for this metric.

On Amazon PC and Amazon Photo MinCutPool failed to converge, even with tuning the orthogonality regularization parameter. We attribute that to extremely uneven structure of these graphs, as popular products are co-purchased with a lot of other items, so the effects discussed in [17, 32] are prohibiting good cuts. This corresponds to high values of d ad $d_{\max}$ in our synthetic scenarios 5 and 6.

Overall, DMoN demonstrates excellent performance on both graph clustering and label correlation, successfully leveraging both graph and attribute information. Both synthetic and real-world experiments prove that DMoN is vastly superior to its counterparts in attributed graph clustering.

6 Conclusion

In this work, we study GNN pooling through the lens of attributed graph clustering. We introduce Deep Modularity Networks (DMoN), an unsupervised objective and realize it with a GNN which can recover high quality clusters. We compare against challenging baselines that baselines that optimize structure (SBM), features (kmeans), or both (DGI+k-means), in addition to a recently proposed state-of-the-art pooling method (MinCutPool).

We explore the limits of GNN clustering methods in terms of both graph and feature signals using synthetic data, where we see that DMoN better leverages structure and attributes than all existing methods. In extensive experiments on real datasets we show that the clusters found by DMoN are more likely to correspond to ground truth labels, and have better properties as illustrated by clustering metrics (e.g. conductance or modularity). We hope that this work will further advancements in unsupervised learning for GNNs as well as attributed graph clustering, allowing further advances in graph learning.

Broader Impact

We believe that better understanding of graph pooling through the lens of attributed graph clustering will allow for the design of more effective graph neural network architectures. Our reformulation encourages explicit grouping of nodes in graphs, which we envision will be useful in domains with naturally occurring community structure, e.g. social networks or information retrieval. Additionally, our M-GCN could serve as the "graph-context" head in semi-supervised architectures [67], allowing for stronger community-based constraints in those systems.

Our work also allows translation of open questions in network science about attributed graph clustering to the GNN domain, potentially enabling powerful new modelling solutions. Specifically, existing combinatorial [65] and bayesian [39] methods for attributed graph clustering use graph features to improve clustering accuracy. [39] shows that using features allows their estimation-based approach to overcome the spectral "detectability" limit for graphs [35], a result that we replicate in this study. Using our model, more ideas from the network science community may be ported to GNN architectures.

References

[1] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *SODA*, 2007.

[2] Nitin Bansal, Xiaohan Chen, and Zhangyang Wang. Can we gain more from orthogonality regularizations in training deep cnns? In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 4266–4276. Curran Associates Inc., 2018.

[3] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and R Devon Hjelm. Mine: mutual information neural estimation. In *ICML*, 2018.

[4] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Spectral clustering with graph neural networks for graph pooling. In *ICML*, 2020.

[5] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Görke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. Maximizing modularity is hard. *arXiv preprint physics/0608255*, 2006.

[6] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 2017.

[7] Irineo Cabreros, Emmanuel Abbe, and Aristotelis Tsirigos. Detecting community structures in hi-c genomic data. In *2016 Annual Conference on Information Science and Systems (CISS)*, pages 584–589. IEEE, 2016.

[8] Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, and Kevin Murphy. Machine learning on graphs: A model and comprehensive taxonomy. *arXiv preprint arXiv:2005.03675*, 2020.

[9] Haochen Chen, Bryan Perozzi, Yifan Hu, and Steven Skiena. Harp: Hierarchical representation learning for networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[10] Haochen Chen, Bryan Perozzi, Yifan Hu, and Steven Skiena. HARP: Hierarchical representation learning for networks. In *AAAI*, 2018.

[11] Stéphan Clémençon, Hector De Arazoza, Fabrice Rossi, and Viet Chi Tran. Hierarchical clustering for graph visualization. *arXiv preprint arXiv:1210.5693*, 2012.

[12] Weiwei Cui, Hong Zhou, Huamin Qu, Pak Chung Wong, and Xiaoming Li. Geometry-based edge clustering for graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1277–1284, 2008.

[13] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, 2016.

[14] Chenhui Deng, Zhiqiang Zhao, Yongyu Wang, Zhiru Zhang, and Zhuo Feng. GraphZoom: A multi-level spectral approach for accurate and scalable graph embedding. In *ICLR*, 2020.

[15] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *NIPS*, 2015.

[16] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

[17] Alessandro Epasto, Silvio Lattanzi, and Renato Paes Leme. Ego-splitting framework: From non-overlapping to overlapping clusters. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 145–154, 2017.

[18] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 1973.

[19] Santo Fortunato and Darko Hric. Community detection in networks: A user guide. *Physics reports*, 2016.

[20] Hongyang Gao and Shuiwang Ji. Graph U-nets. In *ICML*, 2019.

[21] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *ICML*, 2017.

[22] Benjamin H Good, Yves-Alexandre De Montjoye, and Aaron Clauset. Performance of modularity maximization in practical contexts. *Physical Review E*, 81(4):046106, 2010.

[23] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, 2016.

[24] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, 2010.

[25] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *IEEE Data Engineering Bulletin*, 2017.

[26] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *ICLR*, 2019.

[27] Weihua Hu*, Bowen Liu*, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *ICLR*, 2020.

[28] Brian W Kernighan and Shen Lin. An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, 1970.

[29] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[30] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in neural information processing systems*, pages 971–980, 2017.

[31] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *ICML*, 2019.

[32] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. Statistical properties of community structure in large social and information networks. In *Proceedings of the 17th international conference on World Wide Web*, pages 695–704, 2008.

[33] Jiongqian Liang, Saket Gurukar, and Srinivasan Parthasarathy. MILE: A multi-level framework for scalable graph embedding. *arXiv preprint arXiv:1802.09612*, 2018.

[34] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 1982.

[35] Raj Rao Nadakuditi and Mark EJ Newman. Graph spectra and the detectability of community structure in networks. *Physical review letters*, 2012.

[36] Nicolò Navarin, Dinh V Tran, and Alessandro Sperduti. Pre-training graph neural networks with kernels. In *NeurIPS*, 2018.

[37] Mark EJ Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical review E*, 2006.

[38] Mark EJ Newman. Modularity and community structure in networks. *PNAS*, 2006.

[39] Mark EJ Newman and Aaron Clauset. Structure and inference in annotated networks. *Nature Communications*, 7, 2016.

[40] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *ICML*, 2016.

[41] Zizi Papacharissi. The virtual geographies of social networks: a comparative analysis of facebook, linkedin and asmallworld. *New media & society*, 11(1-2):199–220, 2009.

[42] Tiago P Peixoto. Efficient monte carlo and greedy heuristic for the inference of stochastic block models. *Physical Review E*, 2014.

[43] Tiago P Peixoto. *The graph-tool python library*, 2014 (accessed June 1, 2020). http://figshare.com/articles/graph_tool/1164194.

[44] Bryan Perozzi and Leman Akoglu. Scalable anomaly ranking of attributed neighborhoods. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 207–215. SIAM, 2016.

[45] Bryan Perozzi and Leman Akoglu. Discovering communities and anomalies in attributed graphs: Interactive visual exploration and summarization. *ACM Trans. Knowl. Discov. Data*, 12(2), January 2018.

[46] Bryan Perozzi, Leman Akoglu, Patricia Iglesias Sánchez, and Emmanuel Müller. Focused clustering and outlier detection in large attributed graphs. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, page 1346–1355, New York, NY, USA, 2014. Association for Computing Machinery.

[47] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, page 701–710, New York, NY, USA, 2014. Association for Computing Machinery.

[48] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. DeepWalk: Online learning of social representations. In *KDD*, 2014.

[49] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 2008.

[50] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 2008.

[51] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

[52] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 2000.

[53] Tom AB Snijders and Krzysztof Nowicki. Estimation and prediction for stochastic blockmodels for graphs with latent block structure. *Journal of classification*, 1997.

[54] Jiaming Song and Stefano Ermon. Understanding the limitations of variational mutual information estimators. In *ICLR*, 2020.

[55] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 2014.

[56] Fan-Yun Sun, Jordan Hoffmann, and Jian Tang. InfoGraph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *ICLR*, 2020.

[57] Amanda L Traud, Eric D Kelsic, Peter J Mucha, and Mason A Porter. Comparing community structure to characteristics in online collegiate social networks. *SIAM review*, 53(3):526–543, 2011.

[58] Michael Tschannen, Josip Djolonga, Paul K Rubenstein, Sylvain Gelly, and Mario Lucic. On mutual information maximization for representation learning. In *ICLR*, 2020.

[59] Anton Tsitsulin, Davide Mottin, Panagiotis Karras, and Emmanuel Müller. Verse: Versatile graph embeddings from similarity measures. In *WWW*, 2018.

[60] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2017.

[61] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *ICLR*, 2019.

[62] Zhongdao Wang, Liang Zheng, Yali Li, and Shengjin Wang. Linkage based face clustering via graph convolution network. In *CVPR*, 2019.

[63] Yen-Chuen Wei and Chung-Kuan Cheng. Towards efficient hierarchical designs by ratio cut partitioning. In *IEEE International Conference on Computer-Aided Design*. IEEE, 1989.

[64] Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 2015.

[65] Jaewon Yang, Julian McAuley, and Jure Leskovec. Community detection in networks with node attributes. In *2013 IEEE 13th International Conference on Data Mining*, pages 1151–1156. IEEE, 2013.

[66] Lei Yang, Xiaohang Zhan, Dapeng Chen, Junjie Yan, Chen Change Loy, and Dahua Lin. Learning to cluster faces on an affinity graph. In *CVPR*, 2019.

[67] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *ICML*, 2016.

[68] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 974–983, 2018.

[69] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *NeurIPS*, 2018.

[70] Jiaxuan You, Rex Ying, and Jure Leskovec. Position-aware graph neural networks. In *ICML*, 2019.