Community Detection with Graph Neural Networks

Joan Bruna
Courant Institute of Mathematical Sciences and Center for Data Science
New York University
New York, NY, 10012
bruna@cims.nyu.edu
Xiang (Lisha) Li
Statistics Department, UC Berkeley
Berkeley, CA, 94720

May 30, 2017

Abstract

We study data-driven methods for community detection in graphs. This estimation problem is typically formulated in terms of the spectrum of certain operators, as well as via posterior inference under certain probabilistic graphical models. Focusing on random graph families such as the Stochastic Block Model, recent research has unified these two approaches, and identified both statistical and computational signal-to-noise detection thresholds.

We embed the resulting class of algorithms within a generic family of graph neural networks and show that they can reach those detection thresholds in a purely data-driven manner, without access to the underlying generative models and with no parameter assumptions. The resulting model is also tested on real datasets, requiring less computational steps and performing significantly better than rigid parametric models.

1 Introduction

Clustering and community detection is a fundamental unsupervised data analysis task. Given some relational observations between a set of incoming datapoints, it consists in inferring a community structure across the dataset that enables non-linear dimensionality reduction and analysis. Efficient algorithms to perform clustering – such as k-means exist, and are heavily used across diverse data science areas. However, it relies on having the appropriate Euclidean embedding of the data.

By formulating this task as a graph partitioning problem, spectral clustering methods obtain such embedding from the leading eigenvectors of appropriate operators defined on the graph, such as the Normalized Graph Laplacian. This leads to efficient algorithms, yet there is no general procedure to construct the "correct" graph operator from the data. Another formalism is based on Probabilistic Graphical Models. By postulating the community structure as a latent, unobserved variable, authors have constructed latent generative models, using for instance pairwise Markov random fields, where inferring the community structure can be seen as a form of posterior inference over the graphical model. Whereas such models offer flexibility that lacks in spectral clustering models, the Maximum-Likelihood Estimation over such graphical models is in general intractable. However, when the underlying graph has a particular 'tree-like' structure, Belief Propagation (BP) provides a way forward. In the specific instance of the *Stochastic Block Model* (SBM), a recent research program ([1] and references therein) has bridged the gap between probabilistic and spectral methods using tools from Statistical Physics, leading to a rich understanding of statistical and computational estimation limits.

In this work, we study to what extent one can *learn* those algorithms by observing labeled pairs of graphs and labels. In other words, we observe instances of graphs together with their true community structure, and attempt to learn a mapping between graphs and their predicted communities. We propose to do so by unrolling the previous generic inference algorithms and by using backpropagation. Our motivation is both to obtain computationally efficient estimation, and robustness against model misspecifications.

Spectral clustering algorithms consist in performing power iterations, which are also alternating between localized linear operators in the graph, point-wise nonlinearities and normalization. This algorithm can therefore be "unrolled" and recast as a neural network, similarly as in [8]. In our scenario, the resulting neural network for the community detection task is the graph neural network (GNN) [20, 3]. Inspired by the works that assimilate efficient Belief Propagation algorithms with specific perturbations of the spectrum of the Graph Laplacian, we propose key modifications to GNNs that provide us with a robust model that can operate well under scenarios where standard spectral clustering methods fail.

We first study our GNN model in the synthetic Stochastic Block Model, for which the computational and information-theoretic detection regimes and corresponding algorithms are well-known. We show that our network learns to reach those detection thresholds with no explicit knowledge of the model, and with improved computational efficiency. Next, we demonstrate the applicability of our framework on real-world community detection problems, by showing that our data-driven model is able to outperform existing community detection algorithms based on parametric generative families.

To summarize, our main contributions are:

- We propose to use graph neural networks to perform data-driven spectral analysis by unrolling power iterations.
- We show that on the Stochastic Block-Model we reach detection thresholds in a purely data-driven fashion.

- We show how our model can be applied to real-world datasets, leading to state-of-the-art community detection results.

2 Background and Problem Setup

In this section, we describe our problem setup and how it relates to spectral clustering and probabilistic graphical inference.

2.1 Graph Min-Cuts and Spectral Clustering

We consider graphs $G = (V, E)$, modeling a system of $N = |V|$ elements presumed to exhibit some form of community structure. The adjacency matrix A associated with G is the $N \times N$ binary matrix such that $A_{i,j} = 1$ whenever $(i, j) \in E$. We assume for simplicity undirected graphs, yielding symmetric adjacency matrices. The community structure is encoded in a discrete label vector $s : V \to \{1, K\}$ that assigns a community label to each node, and the goal is to estimate s from observing the adjacency matrix (up to global label permutation).

In the setting of binary, associative communities, where $s(i) = \pm 1$, two nodes i, j with $s(i) = s(j)$ are more likely to be connected ($A_{i,j} = 1$) than two nodes i, j with $s(i) \neq s(j)$. Thus a quantity of the form

$$\sum_{i,j} (1 - s(i)s(j))A_{i,j}$$

measures the cost associated with *cutting* the graph between communities encoded by s that we wish to minimize under appropriate constraints [17]. Note that $\sum_{i,j} A_{i,j} = s^T D s$, with $D = \mathrm{diag}(A\mathbf{1})$ (called the degree matrix), so the cut cost can be expressed as a positive semidefinite quadratic form

$$\min_{s(i)=\pm 1} s^T(D - A)s = s^T \Delta s$$

that we wish to minimize. This shows a fundamental connection between the community structure and the spectrum of certain linear operators of the graph, which provides a powerful and stable relaxation of the discrete combinatorial optimization problem of estimating the community labels for each node. In the case of the graph Laplacian $\Delta = D - A$, its eigenvector associated with the smallest eigenvalue is trivial, but its Fiedler vector (the eigenvector associated with the second smallest eigenvalue) reveals important community information of the graph [17] under appropriate conditions, and is associated with the graph conductance [21] under certain normalization schemes.

For a given linear operator $\mathcal{L}(A)$ extracted from the graph (that we assume symmetric), we are thus interested in extracting eigenvectors at the edge of its spectrum. A particularly simple algorithmic framework is given by the power iteration method. Indeed, the

Fiedler vector of $\mathcal{L}(A)$ can be obtained by first extracting the leading eigenvector v of $\tilde{A} = \|\mathcal{L}(A)\|\mathbb{I} - \mathcal{L}(A)$, and then iterating

$$y^{(n)} = \tilde{A} w^{(n-1)} \quad , \; w^{(n)} = \frac{y^{(n)} - \langle y^{(n)}, v\rangle v}{\|y^{(n)} - \langle y^{(n)}, v\rangle v\|} \; .$$

Unrolling power iterations and recasting the resulting model as a trainable neural network is akin to the LISTA [8] sparse coding model, which unrolled iterative proximal splitting algorithms.

Despite the appeal of graph Laplacian spectral approaches, it is well known [13] that these methods fail in sparsely connected graphs. Indeed, in such scenarios, the eigenvectors of graph Laplacians concentrate on nodes with dominant degree, losing their ability to correlate with community structure. In order to overcome this important limitation, authors have resorted to ideas inspired from statistical physics, as explained next.

2.2 Probabilistic Graphical Models and Belief-Propagation

Graphs with labels on nodes and edges can be cast as a graphical model where the aim of clustering is to optimize label agreement. This can be seen as a posterior inference task. If we simply assume the graphical model is a Markov Random Field (MRF) with trivial compatibility functions for cliques greater than 2, the probability of a label configuration σ is given by

$$\mathbb{P}(\sigma) = \frac{1}{\mathcal{Z}} \prod_{i \in V} \phi_i(\sigma_i) \prod_{ij \in E} \psi_{ij}(\sigma_i, \sigma_j). \tag{1}$$

Generally, computing marginals of multivariate discrete distributions is exponentially hard. For instance, in the case of $\mathbb{P}(\sigma_i)$ we are summing over $|X|^{n-1}$ terms (where X is the state space of discrete variables). But if the graph is a tree, we can factorize the MRF more efficiently to compute the marginals in linear time via a dynamic programming method called the sum-product algorithm, also known as belief propagation (BP). An iteration of BP is given by

$$b_{i\to j}(\sigma_i) = \frac{1}{Z_{i\to j}} \phi_i(\sigma_i) \prod_{k \in \delta i n j} \sum_{\sigma_k \in X} \psi_{ik}(\sigma_i, \sigma_k) b_{k \to i}(\sigma_k). \tag{2}$$

The beliefs $(b_{i\to j}(\sigma_i))$ are interpreted as the marginal distributions of σ_i. Fixed points of BP can be used to recover marginals of the MRF above. In the case of the tree, the correspondence is exact: $\mathbb{P}_i(\sigma_i) = b_i(\sigma_i)$. Some sparse graph, like the Stochastic Blockmodel with constant degree[16] are locally similar to trees for such an approximation to be successful. BP approximates the MLE solutions but convergence is not guaranteed in graphs that are not trees. Furthermore, in order to apply BP, we need a generative model and the correct parameters of the model. If unknown, the parameters can be derived using expectation maximization, further adding complexity and instability to the method since iterations may learn parameters for which BP does not converge.

2.3 Non-backtracking operator and Bethe Hessian

The BP equations have a trivial fixed-point where every node takes equal probability in each group. Linearizing the BP equation around this point is equivalent to spectral clustering using the non-backtracking matrix (NB), a matrix defined on the edges of the graph that indicates whether two edges are adjacent and do not coincide. Spectral clustering using NB gives significant improvements over spectral clustering with versions of the Laplacians (L) and the adjacency matrix (A) [13]; High degree fluctuations drown out the signal of the informative eigenvalues in the case of A and L, whereas NB's eigenvalues are confined to a disk in the complex plane, so its eigenvalues corresponding to community structure lay outside the disk and are easily distinguishable.

NB matrices are still not optimal in that they are matrices on the edge set, and are not symmetric (so cannot enjoy tools of numerical linear algebra for symmetric matrices). Recently Saade et al.[19] showed that a spectral method can do as well as BP in this regime, using the Bethe Hessian operator given by $BH(r) := (r^2 - 1)\mathbb{I} - rA + D$ (where r is a scalar value). This is due to a one-to-one correspondence between the fixed points of BP and the stationary points of the Bethe free energy (corresponding Gibbs energy of the Bethe approximation)[25]. The Bethe Hessian is a scaling of the Hessian of the Bethe free energy at an extrema corresponding to the trivial fixed point of BP. Negative eigenvalues of $BH(r)$ correspond to phase transitions in the Ising model where new clusters become identifiable. This all gives theoretical motivation for why $[\mathbb{I}, D, A]$ defined in Section 3 are a good family of generators to do spectral clustering on. In the case of the SBM, they generate the Bethe Hessian which can achieve community detection down to the information theoretic threshold. The GNN is capable of expressing spectral approximations of complicated functions of $[\mathbb{I}, D, A]$, and performing nonlinear power method iterations in order to infer global structure (for instance community structure). Furthermore, unlike belief propagation, the method does not require a generative model, oftentimes requires a lot statistical analysis to motivate and is exposed to model misspecifications when deployed on real data. Instead, our framework finds structure in a data driven way, learning it from the available training data.

3 Multiscale Graph Neural Networks

The Graph Neural Network (GNN), introduced in [20] and later simplified in [14, 6, 22]. is a flexible neural network architecture that is based on local operators on a graph $G = (V, E)$. We start by briefly reviewing the generic GNN architecture, and next describe our modifications to make it suitable to our interests.

Given some input signal $F \in \mathbb{R}^{V \times d}$ on the vertices of G, we consider graph intrinsic linear operators that act locally on this signal: The *degree operator* is the linear map $D : F \mapsto DF$ where $(DF)_i := deg(i) \cdot F_i$, $D(F) = \text{diag}(A\mathbf{1})F$. The *adjacency operator* is the map $A : F \mapsto A(F)$ where $(AF)_i := \sum_{j \sim i} F_j$, with $i \sim j$ iff $(i, j) \in E$. Similarly,

J-th powers of A encode J-hop neighborhoods of each node, and allow us to combine and aggregate local information at different scales. We also allow a channel to broadcast information globally giving GNN the ability to recover average degrees, moments of degrees via $(U(F))_i = \frac{1}{|V|}\sum_j F_j$.

We consider a multiscale gnn layer that receives as input a signal $x^{(k)} \in \mathbb{R}^{V \times d_k}$ and produces $x^{(k+1)} \in \mathbb{R}^{V \times d_{k+1}}$ as

$$\begin{aligned} x^{(k+1)}{}_{i,l} &= \rho\left(\theta_{1,l}^{(k)} x_i + \theta_{2,l}^{(k)}(Dx)_i + \theta_{3,l}^{(k)}(Ux)_i + \sum_{j=0}^{J-1}\theta_{4+j,l}^{(k)}(A^{2^j}x)_i\right)\ ,\ l = 1, \dots d_{k+1}/2\ , \\ x^{(k+1)}{}_{i,l} &= \widetilde{\theta}_{1,l}^{(k)} x_i + \widetilde{\theta}_{2,l}^{(k)}(Dx)_i + \widetilde{\theta}_{3,l}^{(k)}(Ux)_i + \sum_{j=0}^{J-1}\widetilde{\theta}_{4+j,l}^{(k)}(A^{2^j}x)_i\ ,\ l = d_{k+1}/2+1, \dots d_{k+1} \end{aligned} \tag{3}$$

where $\Theta = \{\theta_1^{(k)}, \dots, \theta_{J+3}^{(k)}, \widetilde{\theta}_1^{(k)}, \dots, \widetilde{\theta}_{J+3}^{(k)}\}$, $\theta_s^{(k)} \in \mathbb{R}^{d_{k+1} \times d_k}$ are trainable parameters and $\rho(\cdot)$ is a point-wise nonlinearity, chosen in this work to be $\rho(z) = \max(0, z)$. We thus consider a layer with linear "residual connections" [9], to both ease with the optimization when using large number of layers, but also to give the model the ability to perform power iterations. Since the spectral radius of the learnt linear operators in (3) can grow as the optimization progresses, the cascade of gnn layers can become unstable to training. In order to mitigate this effect, we consider spatial batch normalization [10] at each layer.

As explained in Section 2.1, the Krylov subspace generated by the graph Laplacian [5] is not sufficient in this case to operate well in the sparse regime, as opposed to the generators $\{\mathbb{I}, D, A\}$. The expressive power of each layer is increased by adding multiscale versions of A, although this benefit comes at the cost of computational efficiency, especially in the sparse regime. The network depth is chosen to be of the order of the graph diameter, so that all nodes obtain information from the entire graph. In sparse graphs with small diameter, this architecture offers excellent scalability and computational complexity. Indeed, in many social networks diameters are constant (due to hubs), or $\sim\ \log(|V|)$, as in the Stochastic Block-Model in the constant average degree regime [18]. This results in a model with computational complexity of the order of $\sim |V|\log(|V)$, making it amenable to large-scale graphs.

In our setup, batch normalization not only prevents gradient blowup, but also performs the orthogonalisation relative to the constant vector, which is associated with the smallest eigenvector of the graph operator whose spectrum contains community information. This reinforces the analogy between cascading layers of (3) and the power iterations to obtain the Fiedler vector of such operator. Indeed, if one wants to extract the Fiedler vector of a matrix M, whose smallest eigenvector is known to be v, one can do so by performing power iterations on $\tilde{M} = \|M\|\mathbb{I} - M$ as

$$y^{(n+1)} = \tilde{M}x^{(n)}\ ,\ x^{(n+1)} = \frac{y^{(n+1)} - v^T v y^{(n+1)}}{\|y^{(n+1)} - v^T v y^{(n+1)}\|}\ .$$

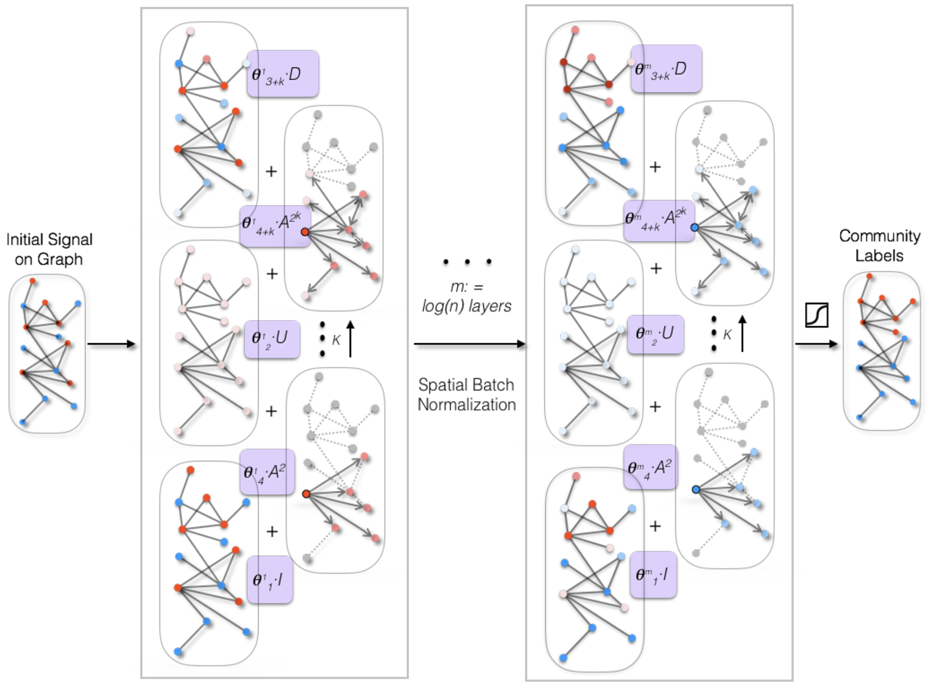

Figure 1: We input an arbitrary signal (can be random or can be informative) and output a classification of the nodes. The colour saturation representative of the magnitude of the signal, whereas the colour difference encode different label classes (red versus blue in this case).

If v is a constant vector, then the normalization above is precisely performed within the Batch Normalization step.

We bootstrap the network by considering the input signal $x^{(0)} = \text{deg}$. After performing K steps of (3), we use the resulting node-level features to predict the community of each node. Let $\mathcal{C} = \{c_1, \ldots, c_C\}$ denote the possible community labelings that each node can take.

Consider first the case where communities do not overlap: C equals the number existing communities. We define the network output at each node using standard softmax, computing the conditional probability that node i belongs to community c: $o_{i,c} = \frac{e^{\langle \theta_c^{(o)}, x_{i,\cdot}^{(K)} \rangle}}{\sum_{c'} e^{\langle \theta_{c'}^{(o)}, x_{i,\cdot}^{(K)} \rangle}}$, $c = 1 \ldots C$. Let $G = (V, E)$ be the input graph and let $y \in \mathcal{C}^V$ be the ground truth community structure. Since community belonging is defined up to global label changes in communities, we define the loss associated with a given graph instance as

$$\ell(\theta) = \inf_{\sigma \in S_{\mathcal{C}}} - \sum_{i \in V} \log o_{i,\sigma(y_i)} \ , \tag{4}$$

where $S_{\mathcal{C}}$ denotes the permutation group of C elements. In our experiments we considered examples with small number of communities $C = 2, 3, 4$, but general scenarios, where C is suspected to be much larger, might make the evaluation of (4) over the permutation group of C elements impractical. Two possible solutions can be considered: first, given the inferred community memberships $o_{i,c}$, we can extract the permutation subgroup of size $\tilde{C}!$ determined by the $\tilde{C}$ true communities where the model produces the highest uncertainty, measured with the entropy $H(o; c) = -\sum_{y_i = c} o_{i,c} \log o_{i,c}$. Another potential alternative can be used when there is a hierarchical community structure, that would allow us to train each level of the hierarchy with (4) separately. Finally, if we are in a setup where nodes can belong to multiple communities, we simply redefine $\mathcal{C}$ to include subsets of communities instead of just singletons, and modify the permutation group $S_{\mathcal{C}}$ accordingly.

4 Related Work

The GNN was first proposed in [20]. [4] generalized convnets to apply to graphs by using the graph Laplacian's eigenbasis. Signals on graphs projected onto this basis will have coefficients that rapidly decay if the graph that signals live on is very local. [5] and [12] both use symmetric Laplacian modules as effective embedding mechanisms for graph signals, and the latter considered the related task of semi-supervised learning on graphs. However this basis restricts the expressive power of its spectral approximations, a limitation that precludes their application to very sparse graphs (for instance near information theoretic threshold of the SBM). [7] interpreted the GNN architecture as learning a message passing algorithm. As mentioned in our setup, that is a natural relaxation of the inference problem on the MRF, but does not tell the whole story with respect to clustering and the need for generators $[I, D, A]$ in the GNN layers. The present work deals with graphs that are far sparser then previous applications of neural networks on graphs. We are able to be competitive with spectral methods specifically designed to work up to the information theoretic regime that cannot be achieved with previous GNN implementations. See [3] for a recent and more exhaustive survey on deep learning on graphs.

[26] works on data regularization for clustering and rank estimation and is also motivated by success of using Bethe-Hessian-like perturbations to improve spectral methods on sparse networks. They find good perturbations via matrix perturbations, and also had success on the Stochastic Block Model. Jure and his coauthors have done a lot of work in community detection in curating benchmark datasets, quantifying the quality of these datasets[24], and also in making new algorithms for community detection by fitting data to newly designed generative models (models that exhibit similar statistical structure learned from their analysis of the aforementioned datasets)[23].

5 SBM

We briefly review the main properties needed in our analysis, and refer the interested reader to [1] for an excellent recent review. The Stochastic Blockmodel (SBM) is a random graph model denoted by $SBM(n, p, q, K)$. Implicitly there is an $F : V \to \{1, ..., K\}$ associated with each SBM graph, which assigns community labels to each vertex. One obtains a graph from this generative model by starting with n vertices and connecting any two vertices u, v independently at random with probability p if $F(v) = F(u)$, and with probability q if $F(v) \neq F(u)$. We say the SBM is *balanced* if the communities are the same size. Let $\bar{F_n} : V \to \{1, ..., K\}$ be our predicted community labels for $SBM(n, p, q.K)$, F_n's give *exact recovery* on a sequence $\{SBM(n, p, q)\}_n$ if $\mathbb{P}(F_n = \bar{F_n}) \to_n 1$, and give *detection* $\exists \epsilon > 0 : \mathbb{P}(|F_n - \bar{F_n}| \geq 1/k + \epsilon) \to_n 1$ (i.e $\bar{F_n}$'s do better than random guessing).

It is harder to tell communities apart if p is close to q (if $p = q$ we just get an Erdős Renyi random graph, which has no communities). In the two community case, It was shown that exact recovery is possible on $SBM(n, p = \frac{a \log n}{n}, q = \frac{b \log n}{n})$ if and only if $\frac{a+b}{2} \geq 1 + \sqrt{ab}$ [15, 2]. For exact recovery to be possible, p, q must grow at least $O(\log n)$ or else the sequence of graphs will not to be connected, and thus vertex labels will be underdetermined. There is no information-computation gap in this regime, so there are polynomial time algorithms when recovery is possible ([1][15]). In the much sparser regime of constant degree SBM $(n, p = \frac{a}{n}, q = \frac{b}{n})$, detection is the best we hope for. The constant degree regime is also of most interest to us for real world applications, as most large datasets have bounded degree and are extremely sparse. It is also a very challenging regime; spectral approaches using the Laplacian in its various (un)normalized forms and the adjacency matrix, as well as SDP methods cannot detect communities in this regime[1] due to large fluctuations in the degree distribution that prevent eigenvectors form concentrating on the clusters.

In the constant degree regime with balanced k communities, the Kesten-Stigum threshold is given by $SNR := (a - b)^2/(k(a + (k + 1)b))$ [1]. It has been shown for $k = 2$ that $SNR = 1$ is both the information theoretic and efficient computational threshold where belief propagation (BP) via a polynomial time algorithms. For $k \geq 4$ a gap emerges between the information theoretic threshold and computational one. It's conjectured that no polynomial time algorithm exist for $SNR < 1$, while a BP algorithm works for $SNR > 1$[1]. The existence of the gap was shown by [1] by proving a non-polynomial algorithm can do detection for some $SNR < 1$.

6 Experiments

6.1 GNN Performance Near Information Theoretic Threshold of SBM

Our performance measure is the overlap between predicted ($\bar{F}$) and true labels (F), which quantifies how much better than random guessing a predicted labelling is. The overlap is given by $\left(\frac{1}{n}\sum_u \delta_{F(u),\bar{F}(u)} - \frac{1}{k}\right)/(1-\frac{1}{k})$ where δ is the Kronecker delta function, and the labels are defined up to global permutation. The GNNs were all trained with 30 layers, 10 feature maps and $J = 3$ in the middle layers and $n = 1000$. We used Adamax [11] with learning rate 0.001 [1] We consider two learning scenarios. In the first scenario, we train parameters θ conditional on a and b, by producing 6000 samples $G \sim SBM(n = 1000, a_i, b_i, k = 2)$ for different pairs (a_i, b_i) and estimating the resulting $\theta(a_i, b_i)$. In the second scenario, we train a single set of parameters θ from a sample of 6000 samples

$$G \sim SBM(n = 1000, a = k\bar{d} - b, b \sim \text{Unif}(0, \bar{d} - \sqrt{\bar{d}}), k = 2)\ ,$$

where the average degree $\bar{d}$ is either fixed constant or also randomized with $\bar{d} \sim \text{Unif}(1, t)$. This training set is very important as it shows our GNN is not just approximating the BH spectral method since the optimal r is not constant in this dataset. Instead, the model's competitive performance in this regime shows that the GNN is able to learn a higher dimensional representation of the optimal r as a function of the data.

Our GNN model is either competitive with BH or beats BH, which achieves the state of the art along with BP, despite not having any access to the underlying generative model (especially in cases where GNN was trained on a mixture of SBM and thus must be able to generalize the r parameter in BH). They all beat by a wide margin spectral clustering methods using the symmetric Laplacian (Ls) and power method (pm) applied to $\|BH\|\mathbb{I} - BH$ using the same number of layers as our model. Thus GNN's ability to predict labels goes beyond approximating spectral decomposition via learning the optimal r for $BH(r)$. The model architecture allows it to learn a higher dimensional function of the optimal perturbation of the multiscale adjacency basis, as well as nonlinear power iterations, that amplify the informative signals in the spectrum; In a data driven way it can generalize the problem without needing to study a generative model.

6.2 Real Datasets from SNAP

We train the GNN on real datasets with community labels provided by SNAP. These datasets have ground truth community labels ranging from social networks to hierarchical co-purchasing networks. Since the data is diverse, performance on these datasets illustrates how well the GNN can learn a data driven spectral approach for each graph. We obtain the training set for the GNN as follows. For each SNAP dataset, we start by focusing only on the 5000 top quality communities provided by the dataset. We then identify edges (i, j) that cross at least two different communities. For each of such edges, we consider the two largest communities C_1, C_2 such that $i \notin C_2$ and $j \notin C_1$, $i \in C_1$, $j \in C_2$, and extract the subgraph determined by $C_1 \cup C_2$, which is connected since all the communities are connected. Finally, we divide the train and test sets by enforcing test examples to contain disjoint communities from those in the training set. In this experiment, due to

[1] Code is publicly available at `https://github.com/joanbruna/GNN_community`

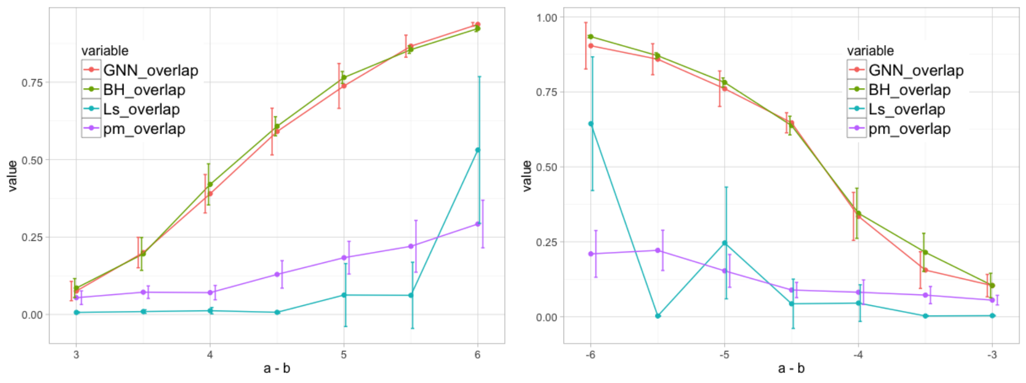

Figure 2: SBM detection. *left:* $k = 2$ associative, *right:* $k = 2$ disasocciative, X-axis corresponds to SNR, Y-axis to overlap; see text.

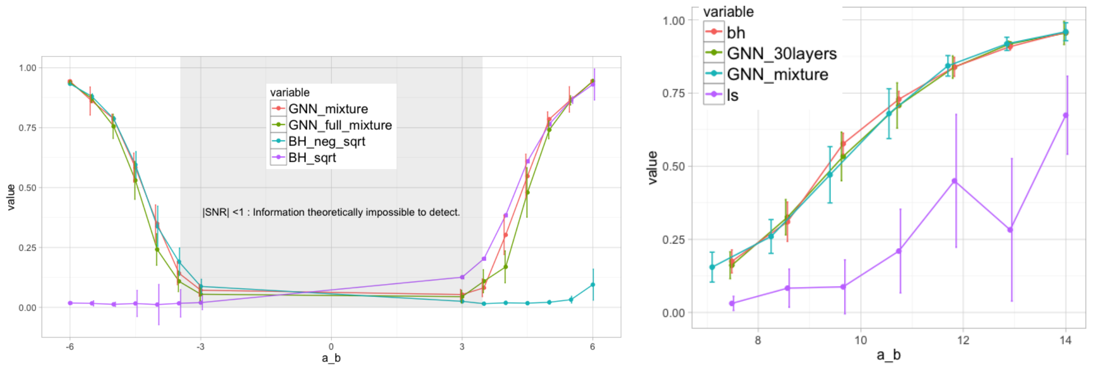

Figure 3: GNN mixture (Graph Neural Network trained on a mixture of SBM with average degree 3), GNN full mixture (GNN trained over different SNR regimes some below threshold), $BH(\sqrt{\bar{d}})$ and $BH(-\sqrt{\bar{d}})$. *left:* $k = 2$, *right:* $k = 4$. We verify that $BH(r)$ models cannot perform detection at both ends of the spectrum simultaneously.

computational limitations, we restrict our attention to the three smallest graphs in the SNAP collection (Youtube, DBLP and Amazon), and we restrict the largest community size to 800 nodes, which is a conservative bound, since the average community size on these graphs is below 30.

We compare GNN's performance with the Community-Affiliation Graph Model (AGM). The AGM is a generative model defined in [23] that allows for overlapping communities where overlapping area have higher density. This was a statistical property observed in many real datasets with ground truth communities, but not present in generative models before AGM and was shown to outperform algorithms before that. AGM fits the data to the model parameters in order give community predictions, and we use the recommended default parameters. Table 1 compares the performance, measured with a 3-class $\{1, 2, 1+2\}$ classification accuracy up to global permutation $1 \leftrightarrow 2$. We stress however that the experimental setup is different from the one in [23], which may impact the performance of AGM. Nonetheless, this experiment illustrates the benefits of data-driven models that strike the right balance between expressive power to adapt to model mis-specifications and structural assumptions of the task at hand.

Table 1: Snap Dataset Performance Comparison between GNN and AGM

	Subgraph Instances			Overlap Comparison	
Dataset	(train/test)	Avg Vertices	Avg Edges	GNN	AGMFit
Amazon	315 / 35	60	346	$\mathbf{0.74 \pm 0.13}$	$\mathbf{0.76 \pm 0.08}$
DBLP	2831 / 510	26	164	$\mathbf{0.78 \pm 0.03}$	0.64 ± 0.01
Youtube	48402 / 7794	61	274	$\mathbf{0.9 \pm 0.02}$	0.57 ± 0.01

7 Conclusion

In this work we have studied data-driven approaches to clustering with graph neural networks. Our results confirm that, even when the signal-to-noise ratio is at the lowest detectable regime, it is possible to backpropagate detection errors through a graph neural network that can 'learn' to extract the spectrum of an appropriate operator. This is made possible by considering generators that span the appropriate family of graph operators that can operate in sparsely connected graphs.

One word of caution is that obviously our results are inherently non-asymptotic, and further work is needed in order to confirm that learning is still possible as $|V|$ grows. Nevertheless, our results open up interesting questions, namely understanding the energy landscape that our model traverses as a function of signal-to-noise ratio; or whether the network parameters can be interpreted mathematically. This could be useful in the study of computational-to-statistical gaps, where our model could be used to inquire about the form of computationally tractable approximations.

Besides such theoretical considerations, we are also interested in pursuing further applications with our model, owing to its good computational complexity $|V|\log(|V|)$. So far it presumes the number of communities to be estimated, so we will explore generalisations to also estimate it to large-scale graphs. Also, our model can readily be applied to data-driven ranking tasks.

References

[1] Emmanuel Abbe. Community detection and stochastic block models: recent developments. *arXiv preprint arXiv:1703.10146*, 2017.

[2] Emmanuel Abbe, Afonso S. Bandeira, and Georgina Hall. Exact recovery in the stochastic block model. *arXiv:1405.3267v4*, 2014.

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

[4] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv:1312.6203.*, 2013.

[5] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proc. NIPS*, 2016.

[6] 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 *Proc. NIPS*, 2015.

[7] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. *arXiv:1704.01212v1*, 2017.

[8] Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. *ICML*, 2010.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[10] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[11] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[12] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[13] Florent Krzakala, Cristopher Moore, Elchanan Mossel, Joe Neeman, Allan Sly, Lenka Zdeborová, and Pan Zhang. Spectral redemption in clustering sparse networks. *Proceedings of the National Academy of Sciences*, 110(52):20935–20940, 2013.

[14] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv:1511.05493*, 2015.

[15] Elchanan Mossel, Joe Neeman, and Allan Sly. A proof of the block model threshold conjecture. *arXiv:1311.4115*, 2014.

[16] Elchanan Mossel, Joe Neeman, and Allan Sly. A proof of the block model threshold conjecture. *arXiv:1311.4115*, 2016.

[17] Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23):8577–8582, 2006.

[18] Oliver Riordan and Nicholas Wormald. The diameter of sparse random graphs. *Combinatorics, Probability and Computing*, 19(5-6):835–926, 2010.

[19] Alaa Saade, Florent Krzakala, and Lenka Zdeborová. Spectral clustering of graphs with the bethe hessian. In *Advances in Neural Information Processing Systems*, pages 406–414, 2014.

[20] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Trans. Neural Networks*, 20(1):61–80, 2009.

[21] Dan Spielman. Spectral graph theory, am 561, cs 662, 2015.

[22] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. *NIPS*, 2016.

[23] Jaewon Yang and Jure Leskovec. Community-affiliation graph model for overlapping network community detection. *Proceeding ICDM '12 Proceedings of the 2012 IEEE 12th International Conference on Data Mining*, 390(.):1170–1175, 2012.

[24] Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. *ICDM.*, 7(2):43–55, dfd.

[25] Jonathan S Yedidia, William T Freeman, and Yair Weiss. Understanding belief propagation and its generalizations. *Exploring artificial intelligence in the new millennium*, 8, 2003.

[26] Pan Zhang. Robust spectral detection of global structures in the data by learning a regularization. *arXiv preprint 1609.02906v1*, 2016.