SUPERVISED COMMUNITY DETECTION WITH LINE GRAPH NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

We study data-driven methods for community detection on graphs, an inverse problem that is typically solved in terms of the spectrum of certain operators or via posterior inference under certain probabilistic graphical models. Focusing on random graph families such as the stochastic block model, recent research has unified both approaches and identified both statistical and computational signal-to-noise detection thresholds.

This graph inference task can be recast as a node-wise graph classification problem, and, as such, computational detection thresholds can be translated in terms of learning within appropriate models. We present a novel family of Graph Neural Networks (GNNs) and show that they can reach those detection thresholds in a purely data-driven manner without access to the underlying generative models, and even improve upon current computational thresholds in hard regimes. For that purpose, we propose to augment GNNs with the non-backtracking operator, defined on the line graph of edge adjacencies. We also perform the first analysis of optimization landscape on using GNNs to solve community detection problems, demonstrating that under certain simplifications and assumptions, the loss value at the local minima is close to the loss value at the global minimum/minima. Finally, the resulting model is also tested on real datasets, performing significantly better than previous models.

1 INTRODUCTION

Graph inference problems encompass a large class of tasks and domains, from posterior inference in probabilistic graphical models to community detection and ranking in generic networks, image segmentation, or graph inverse problems. They are motivated both by practical applications, such as PageRank, but also by fundamental complexity questions, which ask for the intrinsic algorithmic hardness of solving a certain class of graph inference tasks.

These problems can be formulated in either unsupervised, semi-supervised or purely supervised learning settings. In the latter, one assumes a dataset of graphs with labels on its nodes and/or edges, and attempts to perform node/edge classification by optimizing a loss over a certain parametric class, e.g. neural networks. Graph Neural Networks ((Gori et al., 2005), (Bronstein et al., 2017) and references therein) are natural extensions of Convolutional Neural Networks to graph-structured data, and have emerged as a powerful class of algorithms to perform complex graph inference leveraging labeled data. In essence, these neural networks learn cascaded linear combinations of intrinsic graph operators interleaved with node-wise (or edge-wise) activation functions. Since they learn from intrinsic graph operators, they can be applied to varying input graphs, and they offer the same parameter sharing advantages as their CNN counterparts.

In this work, we focus on community detection problems, a wide class of node classification tasks that attempt to discover a clustered, segmented structure within a graph. The algorithmic approaches to this problem include a rich class of spectral methods, which take advantage of the spectrum of certain operators defined on the graph, as well as approximate message-passing methods such as belief propagation (BP), which performs approximate posterior inference under predefined graphical models. Focusing on the supervised setting, we study the ability of GNNs to approximate, generalize or even improve upon these class of algorithms. Our motivation is two-fold. On the one hand, this problem exhibits algorithmic hardness on some settings, opening up the possibility to discover more

efficient algorithms than the current ones. On the other hand, many practical scenarios fall beyond pre-specified probabilistic models, requiring data-driven solutions.

We propose key modifications to the GNN architecture allowing it to exploit edge adjacency information through the non-backtracking operator of the graph. This operator is defined over the edges of the graph and allows a directed flow of information even when the original graph is undirected. We refer to the resulting model as a *Line Graph Neural Network (LGNN)*. Focusing on important random graph families exhibiting community structure, such as the stochastic block model and the geometric block model, we demonstrate improvements in the performance by LGNN, even in regimes within the so-called computational-to-statistical gap. A perhaps surprising aspect is that these gains can be obtained even with *linear* GNNs, which become parametric versions of power iteration algorithms.

This motivates our second main contribution: the analysis of the optimization landscape of such linear GNN models when trained with planted solutions of a given graph distribution. We show that under reparametrization, these landscapes have an interesting property, namely the presence of an *energy gap* controlling the energy difference between local and global minima. With certain assumptions on the spectral concentration of certain random matrices, this energy gap shrinks as the size of the input graphs increases, which would mean that the optimization landscape is benign on large enough graphs.

Summary of Main Contributions:

- We propose an extension of GNNs that operate on the line graph using the non-backtracking operator, which yields significant improvements on hard community detection regimes.
- We show that on the stochastic block model we reach detection thresholds in a purely data-driven fashion and improve upon belief-propogation in hard SBM detection regimes, as well as in the geometric block model.
- We perform the first analysis of the learning landscape of GNN models, showing that under cetain simplifications and assumptions, they exhibit a form of "energy gap", where local mimima are confined in low-energy configurations.
- We show how our model can be applied to real-world datasets, leading to state-of-the-art community detection results.

2 Problem Setup

We are interested in a specific class of node-classification tasks in which given an input graph $G = (V, E)$, a signal $y : V \to \{1, C\}$ encoding a partition of V into C groups is to be predicted at each node. We assume that a training set $\{(G_t, y_t)\}_{t \leq T}$ is given, which we use to learn a model $\hat{y} = \Phi(G, \theta)$ trained by minimising

$$L(\theta) = \frac{1}{T} \sum_{t \leq T} \ell(\Phi(G_t, \theta), y_t) \ .$$

Since y encodes a partition of C groups, the specific label of each node is only important up to a global permutation of $\{1, C\}$. Section 4.3 describes how to construct losses $\ell(a, b)$ with such a property. A permutation of the observed nodes translates into the same permutation applied to the labels, which justifies models Φ that are equivariant to permutations. Also, we are interested in inferring properties of community detection algorithms that do not depend on the specific size of the graphs[1]. We therefore require that the model Φ accepts graphs of variable size for the same set of parameters, similarly as in sequential RNN or spatial CNN models. In our study of random graph models (SBM and GBM), we construct a training set of planted solutions. Labels y_i are generated by sampling a balanced partition uniformly at random, and then we produce the input graphs G_i by sampling $G_i | y_i$ according to each random graph model.

3 Related Work

GNN was first proposed in Gori et al. (2005); Scarselli et al. (2009). Bruna et al. (2013) generalized convolutional neural networks on general undirected graphs by using the graph Laplacian's eigenbasis.

[1] In this work, however, we assume that C is fixed.

This was the first time the Laplacian operator was used in a neural network architecture to perform classification on graph inputs. Defferrard et al. (2016) considers a symmetric Laplacian generator to define a multiscale GNN architecture, demonstrated on classification tasks. Similarly, Kipf & Welling (2016) uses a similar generator as effective embedding mechanisms for graph signals and applies it to semi-supervised tasks. This is the closest application of GNNs to our current contribution. However, we highlight that semi-supervised learning requires bootstrapping the estimation with a subset of labeled nodes, and is mainly interested in generalization within a single, fixed graph. In comparison, our setup considers community detection across a distribution of input graphs and assumes no initial labeling on a given test-set input graph.

There have been several extensions of GNNs (Li et al., 2015; Sukhbaatar et al., 2016; Duvenaud et al., 2015; Niepert et al., 2016) by modifying their non-linear activation functions, their parameter sharing strategies, and their choice of graph operators. In particular, Gilmer et al. (2017) interpreted the GNN architecture as learning an approximate message-passing algorithm, which extends the learning of hidden representations to graph edges in addition to graph nodes. Recently, Velickovic et al. (2017) relates adjacency learning with attention mechanisms, and Vaswani et al. (2017) proposes a similar architecture in the context of machine translation. Another recent and related piece of work is Kondor et al. (2018), which proposes a generalization of GNN that captures high-order node interactions through covariant tensor algebra. Our approach to extend the expressive power of GNN using the line graph may be seen as an alternative to capture such high-order interactions.

Our energy landscape analysis is related to the recent paper (Shamir, 2018), which establishes an energy bound on the local minima arising in the optimization of ResNets. In our case, we exploit the properties of the community detection problem to produce an energy bound that depends on the concentration certain random matrices, which one may hope for as the size of the input graphs increases. Finally, Zhang (2016)'s work on data regularization for clustering and rank estimation is also motivated by the success of using Bethe-Hessian-like perturbations to improve spectral methods on sparse networks. It finds good perturbations via matrix perturbations, and also has success on the stochastic block model. Yang & Leskovec (2012a) curates benchmark datasets for community detection and quantifies the quality of these datasets, while Yang & Leskovec (2012b) develops new algorithms for community detection by fitting data to newly designed generative models, which exhibit similar statistical structure learned from their analysis of the aforementioned datasets.

4 Line Graph Neural Networks

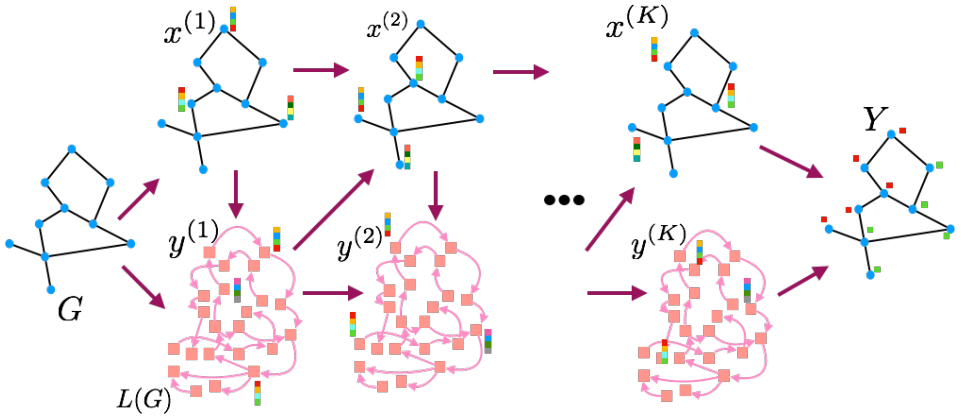

Figure 1. Overview of the architecture of our LGNN. Given an input graph G, we construct its line graph $L(G)$ using the non-backtracking operator (see Figure 2) and we propagate the degree signal through multiple layers of graph diffusion in G and $L(G)$; see equations (1) and (2). The output node features are used to predict node-wise labels, and the whole network is trained end-to-end using standard backpropagation using a label permutation invariant loss (see Section 4.3). The trained model can then be used to infer communities on input graphs of arbitrary size and connectivity.

This section introduces our GNN architectures based on the power graph adjacency (Section 4.1) and its extension to line graphs using the non-backtracking operator (Section 4.2), as well as the design of losses invariant to global label permutations (Section 4.3).

4.1 POWER GRAPH NEURAL NETWORKS

The Graph Neural Network (GNN), introduced in (Scarselli et al., 2009) and later simplified in (GGS; Duvenaud et al., 2015; com) 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 $x \in \mathbb{R}^{|V| \times b}$ 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 $(Dx)_i := deg(i) \cdot x_i$, $D(x) = \text{diag}(A\mathbf{1})x$. The *adjacency operator* A is the linear map given by the adjacency matrix $A_{i,j} = 1$ iff $(i, j) \in E$. In this way, 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 consider in this work the *power graph* adjacency $A_j = \min(1, A^{2^j})$, which encodes 2^j-hop neighborhoods into a binary graph.

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

$$x^{(k+1)}{}_{i,l} = \rho \left[x_i^{(k)} \theta_{1,l}^{(k)} + (Dx^{(k)})_i \theta_{2,l}^{(k)} + \sum_{j=0}^{J-1} (A^{2^j} x^{(k)})_i \theta_{3+j,l}^{(k)} \right], \; l = 1, \ldots b_{k+1}/2, i \in V, \quad (1)$$

$$x^{(k+1)}{}_{i,l} = x_i^{(k)} \theta_{1,l}^{(k)} + (Dx^{(k)})_i \theta_{2,l}^{(k)} + \sum_{j=0}^{J-1} (A^{2^j} x^{(k)})_i \theta_{3+j,l}^{(k)}, \; l = b_{k+1}/2 + 1, \ldots b_{k+1}, i \in V,$$

where $\Theta = \{\theta_1^{(k)}, \ldots, \theta_{J+3}^{(k)}\}$, $\theta_s^{(k)} \in \mathbb{R}^{b_k \times b_{k+1}}$ 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" (He et al., 2016), both to ease with the optimization when using large number of layers and to give the model the ability to perform power iterations. Since the spectral radius of the learned linear operators in (1) 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 (Ioffe & Szegedy, 2015) at each layer.

As explained in Section B.1, the Krylov subspace generated by the graph Laplacian (Defferrard et al., 2016) is not sufficient in this case to operate well in the sparse regime, as opposed to the generators $\{I, D, A\}$. 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 (Riordan & Wormald, 2010). 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 (1) 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 $\tilde{M} = \|M\| I - M$ as $y^{(n+1)} = \tilde{M} x^{(n)}$, $x^{(n+1)} = \frac{y^{(n+1)} - v^T v y^{(n+1)}}{\|y^{(n+1)} - v^T v y^{(n+1)}\|}$. If v is a constant vector, then the normalization above is precisely performed within the batch normalization step.

4.2 LGNN: POWER GNN ON LINE GRAPHS WITH NON-BACKTRACKING OPERATOR

For graphs that have few cycles, posterior inference can be remarkably approximated by loopy belief propagation (Yedidia et al., 2003). As described in Section B.2, the message-passing rules are defined over the edge adjacency graph; see equation 57. Although its second-order approximation

around the critical point can be efficiently approximated with a power method over the original graph, a data-driven version of BP requires accounting for the non-backtracking structure of the message-passing. In this section we describe how to upgrade the GNN model so that it can exploit non-backtracking operators.

The line graph $L(G) = (V_L, E_L)$ is the graph representing the edge adjacency structure of G. If $G = (V, E)$ is an undirected graph, then the vertices V_L of $L(G)$ are the ordered edges in E, that is $V_L = \{(i \to j); (i,j) \in E\} \cup \{(j \to i); (i,j) \in E\}$, so $|V_L| = 2|E|$. The non-backtracking operator $B \in \mathbb{R}^{2|E| \times 2|E|}$ encodes the edge adjacency structure as follows. Two nodes in $L(G)$ are connected if

$$B_{(i\to j),(i'\to j')} = \begin{cases} 1 & \text{if } j = i' \text{ and } j' \neq i\,, \\ 0 & \text{otherwise.} \end{cases}$$

This operator thus enables the propagation of directed information through the graph. The message-passing rules of BP can be expressed as a diffusion in the line graph $L(G)$ using this non-backtracking operator, with specific choices of activation function that turn product of beliefs into sums.

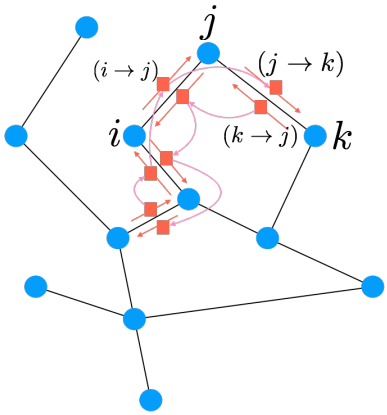

Figure 2. Construction of the line graph $L(G)$ using the non-Backtracking Operator. The nodes of $L(G)$ correspond to oriented edges of G.

A natural extension of the GNN architecture presented in Section 4.1 is thus to consider a second GNN defined on $L(G)$, generated by the corresponding non-backtracking operator B and degree $D_B = \mathrm{diag}(B\mathbf{1})$ operators. This effectively defines *edge features* that are diffused and updated according to the edge adjacency of G. Edge and node features are combined at each layer using the edge indicator matrices $\mathrm{Pm}, \mathrm{Pd} \in \{0,1\}^{|V| \times 2|E|}$, defined as $\mathrm{Pm}_{i,(i\to j)} = 1$, $\mathrm{Pm}_{j,(i\to j)} = 1$, $\mathrm{Pd}_{i,(i\to j)} = 1$, $\mathrm{Pd}_{j,(i\to j)} = -1$ and 0 otherwise. Dropping the skip linear connections for ease of exposition, the resulting model becomes

$$\begin{aligned} {x^{(k+1)}}_{i,l} &= \rho\left[x_i^{(k)}\theta_{1,l}^{(k)} + (Dx^{(k)})_i\theta_{2,l}^{(k)} + \sum_{j=0}^{J-1}(A^{2^j}x^{(k)})_i\theta_{3+j,l}^{(k)} + \{\mathrm{Pm},\mathrm{Pd}\}y^{(k)}\theta_{3+J,l}^{(k)}\right],\ i \in V \\ {y^{(k+1)}}_{i',l'} &= \rho\left[y_{i'}^{(k)}\gamma_{1,l'}^{(k)} + (D_{L(G)}y^{(k)})_{i'}\gamma_{2,l'}^{(k)} + \sum_{j=0}^{J-1}(A_{L(G)}^{2^j}y^{(k)})_{i'}\gamma_{3+j,l'}^{(k)} + [\{\mathrm{Pm},\mathrm{Pd}\}^\top x^{(k+1)}]_{i'}\gamma_{3+J,l'}^{(k)}\right],\ i' \in V_L. \end{aligned} \tag{2}$$

with additional parameters $\{\gamma_1^{(k)}, \ldots, \gamma_{J+3}^{(k)}\}$, $\gamma_s^{(k)} \in \mathbb{R}^{b_k \times b_{k+1}}$. The resulting architecture is named as a *Line Graph Neural Network (LGNN)*.

It can be verified that the resulting model $\Phi(G) := x^{(K)}$ satisfies the permutation equivariance property required for the task: $\Phi(G_\pi) = \Pi\Phi(G)$, where Π is the permutation matrix associated with π. Several authors have proposed combining node and edge feature learning (Gori et al., 2005; Gilmer et al., 2017; Velickovic et al., 2017), although we are not aware of works that considered the edge adjacency structure provided by the line graph and the non-backtracking operator. For graph families with constant average degree $\overline{d}$, the line graph has size $2|E| = \overline{d}|V|$ of the same order, making this model feasible from the computational point of view. The line graph construction can be iterated with $L(L(G)), L(\ldots L(L(G))\ldots)$ to yield a *graph hierarchy*, which would capture high-order interactions between the elements of G. Such hierarchical construction relates to other recent efforts to generalize GNNs (Kondor et al., 2018). In our experiments, we use the input signals $x^{(0)} = \deg(G)$ and $y^{(0)} = \deg(L(G))$ in the line graph version.

Relationship between LGNN and edge feature learning approaches: The GNN on the line graph using the non-backtracking operator can be interpreted as learning *directed* edge features from an *undirected* graph. Indeed, if each node i contains two distinct sets of features $x_s(i)$ and $x_r(i)$, the non-backtracking operator constructs edge features from node features while preserving orientation: For an edge $e = (i,j)$, our model constructs oriented edge features $f_{i\to j} = g(x_s(i), x_r(j))$ and

$f_{j\to i} = g(x_r(i), x_s(j))$ (where g is trainable and not necessarily commutative on its arguments) that are subsequently propagated through the graph. Constructing such local oriented structure is shown to significantly improve performance in the next section. (Battaglia et al., 2016) introduced edge features over directed and typed graphs, but does not discuss the undirected case. (Kearnes et al., 2016; Gilmer et al., 2017) learn edge features on undirected graphs using $f_e = g(x(i), x(j))$ for an edge $e = (i, j)$, where g is now commutative on its arguments. Finally, (Velickovic et al., 2017) learns directed edge features on undirected graphs using stochastic matrices as adjacencies (which are either row or column-normalized).

4.3 A Loss Function Invariant Uner Label permutation

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: $o_{i,c} = p(y_i = c\,|\theta, G)$. 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

$$\ell(\theta) = \inf_{\pi \in S_{\mathcal{C}}} - \sum_{i \in V} \log o_{i,\pi(y_i)} \,, \tag{3}$$

where $S_{\mathcal{C}}$ denotes the permutation group of C elements. In our experiments we considered examples with small number of communities $C \in \{2, 5\}$, but general scenarios, where C is suspected to be much larger, might make the evaluation of (3) over the permutation group of C elements impractical. A possible solution is to randomly partition for each sample $C/\tilde{C}$ labels into $\tilde{C}$ groups, then marginalize the model outputs $o_{i,c}$, $c \leq C$ into $\bar{o}_{i,\bar{c}} = \sum_{c \in \bar{c}} o_{i,c}$ and use $\ell(\theta) = \inf_{\pi \in S_{\tilde{c}}} - \sum_{i \in V} \log \bar{o}_{i,\pi(\bar{y}_i)}$, which only involves a permutation group of size $\tilde{C}!$. 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.

5 Energy Landscape of Linear GNN optimization

As described in the numerical experiments, we found that the GNN models without non-linear activations already provide substantial gains relative to baseline (non-trained) algorithms, by finding suitable generalizations of power iterations. This section studies the optimization landscape resulting from this linear assumption. Despite defining a non-convex objective, we prove that the landscape is 'benign' under certain further simplifications, in the sense that the local minima are confined on sublevel sets of low energy.

For simplicity, we consider only the binary $c = 2$ case where we replace the node-wise binary cross-entropy by the squared cosine distance[2], and we assume a single feature map ($d_k = 1$ for all k), and focus on the power GNN described in Section 4.1 (although our analysis carries equally to describe the line graph version; see remarks below). We also make the simplifying assumption to replace the layer-wise spatial batch normalization by a simpler projection onto the unit ℓ_2 ball (thus we do not remove the mean). Without loss of generality, assume that the input graph G has size n, and denote by $\mathcal{F} = \{A_1, \ldots, A_Q\}$ the family of graph operators appearing in (1). Each layer thus applies an arbitrary polynomial $\sum_{q=1}^{Q} \theta_q^{(k)} A_q$ to the incoming node feature vector $x^{(k)}$. Given an input node vector $w \in \mathbb{R}^n$, the network output can thus be written as

$$\hat{Y} = \frac{e}{\|e\|} \,, \text{ with } e = \left(\prod_{k=1}^{K} \sum_{q \leq Q} \theta_q^{(k)} A_q \right) w \,. \tag{4}$$

We highlight that this multilinear GNN setup is fundamentally different from the multilinear fully-connected neural networks whose landscape is well understood (Kawaguchi, 2016). First, the output is normalized in the sphere, which has important effects in the geometry. Next, the network parametrization is intrinsic (the operators O_j depend on the input), which introduces fluctuations in the

[2]to account for the invariance up to global flip of label

landscape that we analyze. In general, the operators in $\mathcal{F}$ are not commutative, but by considering the generalised Krylov subspace generated by powers of $\mathcal{F}$, $\mathcal{F}^K = \{O_1 = A_1^K, O_2 = A_1 A_2^{K-1}, O_3 = A_1 A_2 A_1^{K-2}, \dots O_{Q^K} = A_Q^K\}$, one can reparametrise (4) as $e = \sum_{j=1}^{Q^K} \beta_j O_j w$ with $\beta \in \mathbb{R}^M$, with $M = Q^K$. Given the target $y \in \mathbb{R}^n$, the loss incurred by each pair (G, y) becomes $\frac{|\langle e, y\rangle|^2}{\|e\|^2}$, and therefore the population loss, when expressed in terms of β, equals

$$L_n(\beta) = \mathbb{E}_{X_n, Y_n} \frac{\beta^\top Y_n \beta}{\beta^\top X_n \beta} \text{ , with} \tag{5}$$

$$Y_n = z_n z_n^\top \in \mathbb{R}^{M\times M} \text{ , } (z_n)_j = \langle O_j w, y\rangle \text{ and } X_n = U_n U_n^\top \in \mathbb{R}^{M\times M} \text{ , } U_n = \begin{bmatrix} (O_1 w)^\top \\ \dots \\ (O_M w)^\top \end{bmatrix} .$$

The landscape is thus specified by a pair of random matrices $Y_n, X_n \in \mathbb{R}^{M\times M}$. The following theorem establishes that under appropriate assumptions, the concentration of certain random matrices around their mean controls the energy gaps between local and global maxima of L.

We define a "mean-field" loss function $\tilde{L}_n(\beta) = \mathbb{E}_{X_n, Y_n} \frac{\beta^T Y_n \beta}{\beta^T \mathbb{E} X_n \beta} = \frac{\beta^T \mathbb{E} Y_n \beta}{\beta^T \mathbb{E} X_n \beta}$, and consider L_n as a perturbation of $\tilde{L}_n$. Assuming that $\mathbb{E}X_n \succ 0$, we write the Cholesky decomposition of $\mathbb{E}X_n$ as $\mathbb{E}X_n = R_n R_n^T$, and define $A_n = R_n^{-1} Y_n (R_n^{-1})^T$, $\bar{A}_n = \mathbb{E}A_n = R_n^{-1}\mathbb{E}Y_n(R_n^{-1})^T$, $B_n = R_n^{-1} X_n (R_n^{-1})^T$, and $\Delta B_n = B_n - I_n$. Given a symmetric matrix $K \in \mathbb{R}^{M\times M}$, we let $\lambda_1(K), \lambda_2(K), ..., \lambda_M(K)$ denote the eigenvalues of K in nondecreasing order.

Theorem 5.1. *For a given n, let $\eta_n = (\lambda_1(\bar{A}_n) - \lambda_2(\bar{A}_n))^{-1}$, $\mu_n = \mathbb{E}[|\lambda_1(A_n)|^6]$, $\nu_n = \mathbb{E}[|\lambda_1(B_n)|^{-6}]$, $\delta_n = \mathbb{E}[\|\Delta B_n\|^6]$, and assume that all four quantities are finite. Then if $\beta_l \in \mathbb{S}^{M-1}$ is a local minimum of L_n, and $\beta_g \in \mathbb{S}^{M-1}$ is a global minimum of L_n, we have $L_n(\beta_l) \geq (1 - \epsilon_{\eta_n,\mu_n,\nu_n,\delta_n}) \cdot L_n(\beta_g)$, where $\epsilon_{\eta_n,\mu_n,\nu_n,\delta_n} = O(\delta_n)$ for given η_n, μ_n, ν_n as $\delta_n \to 0$ and its formula is given in the appendix.*

Corollary 5.2. *If $(\eta_n)_{n\in\mathbb{N}^*}$, $(\mu_n)_{n\in\mathbb{N}^*}$, $(\nu_n)_{n\in\mathbb{N}^*}$ are all bounded sequences, and $\lim_{n\to\infty} \delta_n = 0$, then $\forall \epsilon > 0$, $\exists n_\epsilon$ such that $\forall n > n_\epsilon$, $|L_n(\beta_l) - L_n(\beta_g)| \leq \epsilon \cdot L_n(\beta_g)$.*

The main strategy of the proof is to consider the actual loss function L_n as a perturbation of $\tilde{L}_n$, which has a landscape that is easier to analyze and does not have poor local maxima, since it is equivalent to a quadratic form defined over the sphere $\mathbb{S}^{M-1}$. For a given graph inverse problem, this theorem thus requires estimating spectral fluctuations of the pair X_n, Y_n, which in turn involve the spectrum of C^* algebras generated by the non-commutative family $\mathcal{F}$. That said, one should expect concentration to happen in general, since the dimension M is fixed as n grows. Another interesting question is to understand how the asymptotics of our landscape analysis relate to the hardness of estimation as a function of the Signal-to-Noise ratio. Finally, another open question is to what extent our result could be extended to the non-linear residual GNN case, perhaps leveraging ideas from (Shamir, 2018).

6 EXPERIMENTS

We present experiments on synthetic community detection (Sections 6.1, 6.2 and Appendix C) as well as real-world detection (Section 6.3). In the synthetic experiments, our performance measure is the overlap between predicted ($\hat{y}$) and true labels (y), which quantifies how much better than random guessing a predicted labelling is. The overlap is given by $\left(\frac{1}{n}\sum_u \delta_{y(u),\hat{y}(u)} - \frac{1}{C}\right)/(1 - \frac{1}{C})$ where δ is the Kronecker delta function, and the labels are defined up to global permutation. The GNNs were all trained with 30 layers, 2 feature maps and $J = 2$. We used Adamax (Kingma & Ba, 2014) with learning rate 0.004 across all experiments. [3]

6.1 BINARY STOCHASTIC BLOCK MODEL

The stochastic block model is a random graph model with planted community structure. In its simplest form, one assigns $|V| = n$ nodes to C classes at random with $y : V \to \{1, C\}$ and draws an

[3] Code will be publicly released at ???.???.???

GNN	LGNN	LGNN linear	GAT (Velickovic et al., 2017)	BP
0.17 ± 0.012	$\mathbf{0.207 \pm 0.015}$	0.165 ± 0.015	0.164 ± 0.047	0.1435 ± 0.02

Table 1: Performance of different models on 5-community dissociative SBM graphs with $n = 400$, $C = 5$, $p = 0$, $q = 18/n$, corresponding to average degree $\overline{d} = 14.5$.

edge connecting any two vertices u, v independently at random with probability p if $y(v) = y(u)$, and with probability q otherwise. The sparse binary case $C = 2$ when $p, q \simeq 1/n$ is well understood and provides an initial platform to compare the GNN against provably optimal recovery algorithms; see Appendix B. We consider two learning scenarios. In the first scenario, we train parameters θ conditional on p and q, by producing 6000 samples $G \sim SBM(n = 1000, p_i, p_i, C = 2)$ for different pairs (p_i, q_i) and estimating the resulting $\theta(p_i, q_i)$. In the second scenario, reported in Appendix D, we train a single set of parameters θ from a sample of 6000 samples containing a mixture of SBM with different parameters p, q and average degree. This setup is important as it shows our GNN is not just approximating known algorithms such as BP, since the parameters are not constant in this dataset. Figure 3 reports the performance of our models on the binary SBM model for different SNR regimes and compares it with the belief-propagation baseline from (Decelle et al., 2011), as well as the baseline spectral method using the normalized Laplacian. We observe that our models reach the statistical detection threshold, given in this case by the BP algorithm. Notice that even the linear GNN matches the performance, in accordance to the spectral approximations of BP given by the Bethe Hessian (see supplementary), and significantly outperforms performing 30 power iterations on that operator. We notice the line-graph version of our GNN slightly outperforms the baseline GNN, and that even the linear model that only considers residual connections reaches the statistical threshold. We also notice that our models outperform the Graph Attention Network (GAT) [4] in this task, which we also set to have 30 layers and 2 feature maps (Velickovic et al., 2017). A possible reason is that our graph operators include the degree matrix, which is important in sparse graphs to prevent hub nodes from dominating the diffusion. We ran experiments in the disassociative case ($q > p$), as well as with $C = 3$ communities, and obtained similar results, not reported here.

6.2 Computational-to-Statistical Thresholds in the SBM

The previous section showed that for small number of communities ($k < 4$), the GNN-based model is able to reach the information theoretic (IT) threshold. In such regimes, it is known (Abbe, 2017; Massoulié, 2014; Coja-Oghlan et al., 2016) that BP provably reaches such IT threshold. The situation is different for $k > 4$, where it is conjectured that a computational-to-statistical gap emerges between the theoretical performance of MLE estimators and any polynomial-time estimation procedure (Decelle et al., 2011). In this context, one can use the GNN model to search the space of BP generalizations, and attempt to improve the detection performance of BP for signal-to-noise ratios falling within the computational-to-statistical gap. Table 1 presents results for the 5-community disassociative case, with $p = 0$ and $q = 18/n$, and $n = 400$. This amounts to solving a graph coloring problem in a sparse regime, which falls above the IT threshold but below the regime where BP is able to detect (Decelle et al., 2011), asymptotically as $n \to \infty$. We see that the GNN models significantly outperform BP in this regime, and that the line GNN version provides the best overlap performance, opening up the possibility to reduce the computation-information gap. That said, our model may be picking finite-size effects, which may vanish as $n \to \infty$; the asymptotic study of these gains is left for future work.

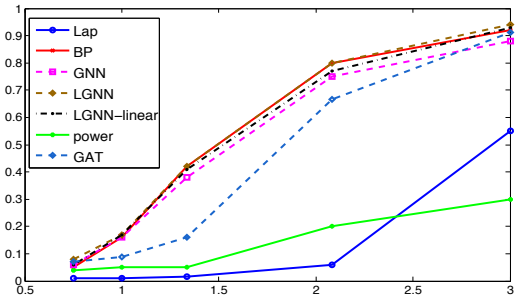

Figure 3. SBM detection. $C = 2$ associative, X-axis corresponds to SNR, Y-axis to overlap; see text.

[4] We based our implementation from `https://github.com/Diego999/pyGAT`. We modified the code so that the number of layers in the network is flexible, and also added spatial batch normalization at the end of each layer. Our experiments showed that including spatial batch normalization improves the performance.

6.3 REAL DATASETS FROM SNAP

We now train the GNNs 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. We obtain the training set 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 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 our GNN's performance with the Community-Affiliation Graph Model (AGM) and with a variant of the LGNN that considers symmetric edge features instead of the non-backtracking operator, which fits into the framework of MPNNs (Gilmer et al., 2017) using ReLU activations. The AGM is a generative model defined in Yang & Leskovec (2012b) that allows 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. Table 2 compares the performance, measured with a 3-class $\{1, 2, 1 + 2\}$ classification accuracy up to global permutation $1 \leftrightarrow 2$. It illustrates the benefits of data-driven models that strike the right balance between expressive power to adapt to model misspecifications and structural assumptions of the task at hand.

Table 2: Snap Dataset Comparison between GNN and AGM. We report node classification accuracy. We compare against our implementation of MPNNs based on symmetric edge adjacencies (see text).

Dataset	(train/test)	Avg $\lvert V\rvert$	Avg $\lvert E\rvert$	GNN	LGNN	MPNN*	AGMFit
Amazon	268 / 52	60	346	0.78 ± 0.13	$\mathbf{0.96 \pm 0.1}$	0.93 ± 0.2	0.81 ± 0.08
DBLP	2831 / 510	26	164	0.85 ± 0.03	$\mathbf{0.87 \pm 0.04}$	$\mathbf{0.86 \pm 0.04}$	0.64 ± 0.01
Youtube	48402 / 7794	61	274	0.86 ± 0.02	$\mathbf{0.89 \pm 0.02}$	0.87 ± 0.02	0.57 ± 0.01

7 CONCLUSION

In this work we have studied data-driven approaches to community detection 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 appropriate operators. This is made possible by considering a family of graph operators that work effectively in sparsely connected graphs, in particular by considering a hierarchical extension that uses the non-backtracking operator in the line graph. We also provide a theoretical analysis of the optimization landscapes in the linearized regime, which shows an interesting transition from rugged to simple as the size of the graphs increase under appropriate concentration conditions.

One word of caution is that our empirical results are inherently non-asymptotic. Whereas models trained for given graph sizes can be used for inference on arbitrarily sized graphs (owing to the parameter sharing of GNNs), further work is needed in order to understand the generalisation properties as $|V|$ increases. Nevertheless, we believe our work opens up interesting questions, namely better understanding how our results on the energy landscape depend upon specific signal-to-noise ratios, or whether the network parameters can be interpreted mathematically. This could be useful in the study of computational-to-statistical gaps, where our model can be used to inquire about the form of computationally tractable approximations. Other directions of future research include the extension to the case where the number of communities is unknown and variable, and potentially increasing with $|V|$, as well as applications to ranking and edge-cut problems.

REFERENCES

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

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

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems*, pp. 4502–4510, 2016.

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

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

Amin Coja-Oghlan, Florent Krzakala, Will Perkins, and Lenka Zdeborova. Information-theoretic thresholds from the cavity method. *arXiv preprint arXiv:1611.00814*, 2016.

Aurelien Decelle, Florent Krzakala, Cristopher Moore, and Lenka Zdeborová. Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. *Physical Review E*, 84(6):066106, 2011.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pp. 3837–3845, 2016.

David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Neural Information Processing Systems*, 2015.

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

M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *Proc. IJCNN*, 2005.

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

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*, pp. 770–778, 2016.

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

Kenji Kawaguchi. Deep learning without poor local minima. In *Advances in Neural Information Processing Systems*, pp. 586–594, 2016.

Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8): 595–608, 2016.

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

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

Risi Kondor, Hy Truong Son, Horace Pan, Brandon Anderson, and Shubhendu Trivedi. Covariant compositional networks for learning graphs. *arXiv preprint arXiv:1801.02144*, 2018.

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.

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

Laurent Massoulié. Community detection thresholds and the weak ramanujan property. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pp. 694–703. ACM, 2014.

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

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

Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pp. 2014–2023, 2016.

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

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

Abishek Sankararaman and François Baccelli. Community detection on euclidean random graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 2181–2200. SIAM, 2018.

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.

Ohad Shamir. Are resnets provably better than linear predictors? *arXiv preprint arXiv:1804.06739*, 2018.

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

Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, pp. 2244–2252, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. *ICDM.*, 7(2):43–55, 2012a.

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, 2012b.

Jonathan S Yedidia, William T Freeman, and Yair Weiss. Understanding belief propagation and its generalizations. *Exploring artificial intelligence in the new millennium*, 8:236–239, 2003.

Pan Zhang. Robust spectral detection of global structures in the data by learning a regularization. In *Arxiv preprint*, pp. 541–549, 2016.

A PROOF OF THEOREM 5.1

We recall the notations used in Theorem 5.1: $\tilde{L}_n(\beta) = \mathbb{E}_{X_n,Y_n} \frac{\beta^T Y_n \beta}{\beta^T \mathbb{E} X_n \beta} = \frac{\beta^T \mathbb{E} Y_n \beta}{\beta^T \mathbb{E} X_n \beta}$ is the mean-field loss function. We write the Cholesky decomposition of $\mathbb{E}X_n$ as $\mathbb{E}X_n = R_n R_n^T$, and define $A_n = R_n^{-1} Y_n (R_n^{-1})^T$, $\bar{A}_n = \mathbb{E}A_n = R_n^{-1}\mathbb{E}Y_n(R_n^{-1})^T$, $B_n = R_n^{-1} X_n (R_n^{-1})^T$, and $\Delta B_n = B_n - I_n$. Given a symmetric matrix $K \in \mathbb{R}^{M\times M}$, we let $\lambda_1(K), \lambda_2(K), ..., \lambda_M(K)$ denote the eigenvalues of K in nondecreasing order.

First, we have

$$|L_n(\beta_l) - L_n(\beta_g)| \leq |L_n(\beta_l) - \tilde{L}_n(\beta_l)| + |\tilde{L}_n(\beta_l) - \tilde{L}_n(\beta_g)| + |\tilde{L}_n(\beta_g) - L_n(\beta_g)| \quad (6)$$

Let us denote by $\tilde{\beta}_g$ a global minimum of the mean-field loss $\tilde{L}_n$. Taking a step further, we can extend this bound to the following one (the difference is in the second term on the right hand side):

Lemma A.1.

$$|L_n(\beta_l) - L_n(\beta_g)| \leq |L_n(\beta_l) - \tilde{L}_n(\beta_l)| + |\tilde{L}_n(\beta_l) - \tilde{L}_n(\tilde{\beta}_g)| + |\tilde{L}_n(\beta_g) - L_n(\beta_g)| \quad (7)$$

Proof of Lemma A.1. We consider two separate cases: The first case is when $\tilde{L}_n(\beta_l) \geq \tilde{L}_n(\beta_g)$. Then $\tilde{L}_n(\beta_l) - \tilde{L}_n(\tilde{\beta}_g) \geq \tilde{L}_n(\beta_l) - \tilde{L}_n(\beta_g) \geq 0$, and so $|L_n(\beta_l) - L_n(\beta_g)| \leq |L_n(\beta_l) - \tilde{L}_n(\beta_l)| + |\tilde{L}_n(\beta_l) - \tilde{L}_n(\tilde{\beta}_g)| + |\tilde{L}_n(\beta_g) - L_n(\beta_g)|$.

The other case is when $\tilde{L}_n(\beta_l) < \tilde{L}_n(\beta_g)$. Note that $L_n(\beta_l) \geq L_n(\beta_g)$. Then $|L_n(\beta_l) - L_n(\beta_g)| \leq |L_n(\beta_l) - \tilde{L}_n(\beta_l)| + |\tilde{L}_n(\beta_g) - L_n(\beta_g)| \leq |L_n(\beta_l) - \tilde{L}_n(\beta_l)| + |\tilde{L}_n(\beta_l) - \tilde{L}_n(\tilde{\beta}_g)| + |\tilde{L}_n(\beta_g) - L_n(\beta_g)|$.

□

Hence, to bound the "energy gap" $|L_n(\beta_l) - L_n(\beta_g)|$, if suffices to bound the three terms on the right hand side of Lemma A.1 separately. First, we consider the second term, $|\tilde{L}_n(\beta_l) - \tilde{L}_n(\tilde{\beta}_g)|$.

Let $\gamma_l = R_n^T \beta_l, \gamma_g = R_n^T \beta_g$ and $\tilde{\gamma}_g = R_n^T \tilde{\beta}_g$. Define $S_n(\gamma) = L_n(R_n^{-T}\gamma)$ and $\tilde{S}_n(\gamma) = \tilde{L}_n(R_n^{-T}\gamma)$, for any $\gamma \in \mathbb{R}^M$. Thus, we apply a change-of-variable and try to bound $|\tilde{S}_n(\gamma_l) - \tilde{S}_n(\tilde{\gamma}_g)|$.

Since β_l is a local maximum of L_n, $\lambda_1(\nabla^2 L_n(\beta_l)) \leq 0$. Since $\nabla^2 S_n(\gamma_l) = R_n^{-1}\nabla^2 L_n(\beta_l) R_n^{-T}$, where R_n is invertible, we know that $\lambda_1(\nabla^2 S_n(\gamma_l)) \leq 0$, thanks to the following lemma:

Lemma A.2. *If $R, Q \in \mathbb{R}^{M\times M}$, R is invertible, Q is symmetric and $\lambda_q > 0$ is an eigenvalue of Q, then $\lambda_1(RQR^T) \geq \lambda \cdot \lambda_M(RR^T)$*

Proof of Lemma A.2. Say $Qw = \lambda w$ for some vector $w \in \mathbb{R}^M$. Let $v = R^{-T}w$. Then $v^T(RQR^T)v = w^T Q w = \lambda \|w\|^2$. Note that $\|w\|^2 = v^T R R^T v \geq \|v\|^2 \lambda_M(RR^T)$. Hence $\lambda_1(RQR^T) \geq \frac{v^T(RQR^T)v}{\|v\|^2} \geq \frac{\lambda\|w\|^2}{\|w\|^2/\lambda_M(RR^T)} \geq \lambda \cdot \lambda_M(RR^T)$ □

Since $\nabla^2 S_n(\gamma_l) = \nabla^2 \tilde{S}_n(\gamma_l) + (\nabla^2 S_n(\gamma_l) - \nabla^2 \tilde{S}_n(\gamma_l))$, there is $0 \geq \lambda_1(\nabla^2 S_n(\gamma_l)) \geq \lambda_1(\nabla^2 \tilde{S}_n(\gamma_l)) - \|\nabla^2 S_n(\gamma_l) - \nabla^2 \tilde{S}_n(\gamma_l)\|$. Hence,

$$\lambda_1(\nabla^2 \tilde{S}_n(\gamma_l)) \leq \|\nabla^2 S_n(\gamma_l) - \nabla^2 \tilde{S}_n(\gamma_l)\| \quad (8)$$

Next, we relate the left hand side of the inequality above to $\cos(\gamma_l, \tilde{\gamma}_g)$, thereby obtaining an upper bound on $[1 - \cos^2(\gamma_l, \tilde{\gamma}_g)]$, which will then be used to bound $|\tilde{S}_n(\gamma_l) - \tilde{S}_n(\tilde{\gamma}_g)|$.

Lemma A.3. $\forall \gamma \in \mathbb{R}^d$,

$$\lambda_1(\nabla^2 \tilde{S}_n(\gamma)) \geq \frac{2}{\|\gamma\|^2}\{[1 - \cos^2(\gamma, \tilde{\gamma}_g)] \cdot [\lambda_1(\bar{A}_n) - \lambda_2(\bar{A}_n)] - 2\|\gamma\| \cdot \|\nabla \tilde{S}_n(\gamma)\|\}$$

Proof of Lemma A.3.

$$\begin{aligned}\nabla^2 \tilde{S}_n(\gamma) =& 2\mathbb{E}\left[\frac{(\gamma^T\gamma)A_n - (\gamma^T A_n\gamma)I}{(\gamma^T\gamma)^2} + \frac{4(\gamma^T A_n\gamma)\gamma\gamma^T - 4(\gamma^T\gamma)A_n\gamma\gamma^T}{(\gamma^T\gamma)^3}\right] \\ =& 2\mathbb{E}\left[\frac{(\gamma^T\gamma)A_n - (\gamma^T A_n\gamma)I}{(\gamma^T\gamma)^2} + \frac{4[(\gamma^T\gamma)A_n - (\gamma^T A_n\gamma)I]\gamma\gamma^T}{(\gamma^T\gamma)^3}\right] \\ =& 2\left[\frac{(\gamma^T\gamma)\bar{A}_n - (\gamma^T \bar{A}_n\gamma)I}{(\gamma^T\gamma)^2} + \frac{4[(\gamma^T\gamma)\bar{A}_n - (\gamma^T \bar{A}_n\gamma)I]\gamma\gamma^T}{(\gamma^T\gamma)^3}\right]\end{aligned} \tag{9}$$

Thus, if we define $Q_1 = (\gamma^T\gamma)[(\gamma^T\gamma)\bar{A}_n - (\gamma^T\bar{A}_n\gamma)I]$, $Q_2 = 4[(\gamma^T\gamma)\bar{A}_n - (\gamma^T\bar{A}_n\gamma)I]\gamma\gamma^T$, we have

$$\nabla^2 \tilde{S}_n(\gamma) = \frac{2}{\|\gamma\|^6}(Q_1 - Q_2) \tag{10}$$

To bound $\lambda_1(\nabla^2 \tilde{S}_n(\gamma))$, we bound $\lambda_1(Q_1)$ and $\|Q_2\|$ as follows:

Since $\bar{A}_n$ is symmetric, let $\hat{\gamma}_1, \dots \hat{\gamma}_M$ be the orthonormal eigenvectors of $\bar{A}_n$ corresponding to nonincreasing eigenvalues $l_1, \dots l_M$. Note that the global minimum satisfies $\tilde{\gamma}_g = \pm\hat{\gamma}_1$. Write $\gamma = \sum_{i=1}^M \alpha_i\hat{\gamma}_i$, and let $\bar{\alpha}_i = \frac{\alpha_i}{\sqrt{\sum_{i=1}^M \alpha_i^2}}$. Then $|\cos(\gamma, \tilde{\gamma}_g)| = |\cos(\gamma, \hat{\gamma}_1)| = |\bar{\alpha}_1|$.

Then,

$$\begin{aligned}\lambda_1(Q_1) =& (\gamma^T\gamma)\left[l_1\sum_{i=1}^M \alpha_i^2 - \sum_{i=1}^M l_i\alpha_i^2\right] \\ \geq& (\gamma^T\gamma)\left[\left((\sum_{i=1}^M \alpha_i^2) - \alpha_1^2\right)(l_1 - l_2)\right] \\ =& (\gamma^T\gamma)^2[(1-\bar{\alpha}_1^2)(l_1 - l_2)]\end{aligned} \tag{11}$$

To bound $\|Q_2\|$:

$$[(\gamma^T\gamma)\bar{A}_n - (\gamma^T\bar{A}_n\gamma)I]\gamma = \sum_{k=1}^M\left[l_k\sum_{i=1}^M \alpha_i^2 - \sum_{i=1}^M l_i\alpha_i^2\right]\alpha_k\hat{\gamma}_k \tag{12}$$

Note that given vectors $v, w \in \mathbb{R}^M$,

$$\|v \cdot w^T\| = |v^T w|$$

Therefore,

$$\begin{aligned}\|Q_2\| =& 4\left|\left(\sum_{k=1}^M \alpha_k\hat{\gamma}_k\right)^T\left(\sum_{k=1}^M[l_k(\sum_{i=1}^M \alpha_i^2) - (\sum_{i=1}^M l_i\alpha_i^2)]\alpha_k\hat{\gamma}_k\right)\right| \\ =& 4\left|\frac{(\gamma^T\gamma)^2}{2}\gamma^T\nabla\tilde{S}(\gamma)\right| \\ \leq& 2(\gamma^T\gamma)^2\|\gamma\|\|\nabla\tilde{S}(\gamma)\|\end{aligned} \tag{13}$$

Thus,

$$\begin{aligned}\lambda_1(Q_1 - Q_2) \geq& \lambda_1(Q_1) - \|Q_2\| \\ \geq& (\gamma^T\gamma)^2([(1-\bar{\alpha}_1^2)(l_1 - l_2)] - 2\|\gamma\|\|\nabla_\gamma S(\gamma)\|)\end{aligned} \tag{14}$$

This yields the desired lemma. □

Combining inequality 8 and Lemma A.3, we get

$$1 - \cos^2(\gamma_l, \tilde{\gamma}_g) \leq \frac{2\|\gamma_l\| \cdot \|\nabla\tilde{S}_n(\gamma_l)\| + \frac{\|\gamma_l\|^2}{2}\|\nabla^2 S_n(\gamma_l) - \nabla^2\tilde{S}_n(\gamma_l)\|}{\lambda_1(\bar{A}_n) - \lambda_2(\bar{A}_n)} \tag{15}$$

Thus, to bound the angle between γ_l and $\tilde{\gamma}_g$, we can aim to bound $\|\nabla\tilde{S}_n(\gamma_l)\|$ and $\|\nabla^2 S_n(\gamma_l) - \nabla^2\tilde{S}_n(\gamma_l)\|$ as functions of the quantities μ_n, ν_n and δ_n.

Lemma A.4.

$$\|\gamma_l\| \cdot \|\nabla \tilde{S}_n(\gamma_l)\| \leq 2\mu_n \nu_n \delta_n (1 + 3\nu_n + \delta\nu_n) \tag{16}$$

Proof of Lemma A.4.

$$\nabla S_n(\gamma) = 2\mathbb{E}\frac{A_n\gamma}{\gamma^T B_n \gamma} - 2\mathbb{E}\frac{(\gamma^T A_n \gamma)B_n\gamma}{(\gamma^T B_n \gamma)^2} \tag{17}$$

$$\nabla \tilde{S}_n(\gamma) = 2\mathbb{E}\frac{A_n\gamma}{\gamma^T \gamma} - 2\mathbb{E}\frac{(\gamma^T A_n \gamma)\gamma}{(\gamma^T \gamma)^2} \tag{18}$$

Combining equations 17 and 18, we get

$$\nabla S_n(\gamma) - \nabla \tilde{S}_n(\gamma) = \mathbb{E}\left[\frac{2(\gamma^T\gamma - \gamma^T B_n \gamma)A_n\gamma}{(\gamma^T B_n\gamma)(\gamma^T\gamma)} - \frac{2(\gamma^T A_n\gamma)[(\gamma^T\gamma)^2 B_n\gamma - (\gamma^T B_n\gamma)^2\gamma]}{(\gamma^T B_n\gamma)^2(\gamma^T\gamma)^2}\right] \tag{19}$$

Since $\nabla S_n(\gamma_l) = 0$, we have

$$\begin{aligned}\|\nabla \tilde{S}_n(\gamma_l)\| =& \left\|\mathbb{E}\left[\frac{2(\gamma_l^T\gamma_l - \gamma_l^T B_n \gamma_l)A_n\gamma_l}{(\gamma_l^T B_n\gamma_l)(\gamma_l^T\gamma_l)} - \frac{2(\gamma_l^T A_n\gamma_l)[(\gamma_l^T\gamma_l)^2 B_n\gamma_l - (\gamma_l^T B_n\gamma_l)^2\gamma_l]}{(\gamma_l^T B_n\gamma_l)^2(\gamma_l^T\gamma_l)^2}\right]\right\| \\ \leq& \frac{2}{\|\gamma_l\|}\mathbb{E}\left[\frac{|\lambda_1(A_n)|\|\Delta B_n\|}{|\lambda_M(B_n)|} + 3\frac{|\lambda_1(A_n)|\|\Delta B_n\|}{\lambda_M^2(B_n)} + \frac{|\lambda_1(A_n)|\|\Delta B_n\|^2}{\lambda_M^2(B_n)}\right]\end{aligned} \tag{20}$$

Then, by the generalized Hölder's inequality,

$$\begin{aligned}\|\nabla \tilde{S}_n(\gamma_l)\| \leq& \frac{2}{\|\gamma_l\|}\Bigg[\left(\mathbb{E}|\lambda_1(A_n)|^3\mathbb{E}\|\Delta B_n\|^3\mathbb{E}\frac{1}{|\lambda_M(B_n)|^3}\right)^{\frac{1}{3}} \\ &+ 3\left(\mathbb{E}|\lambda_1(A_n)|^3\mathbb{E}\|\Delta B_n\|^3\mathbb{E}\frac{1}{|\lambda_M(B_n)|^6}\right)^{\frac{1}{3}} \\ &+ \left(\mathbb{E}|\lambda_1(A_n)|^3\mathbb{E}\|\Delta B_n\|^6\mathbb{E}\frac{1}{|\lambda_M(B_n)|^6}\right)^{\frac{1}{3}}\Bigg].\end{aligned} \tag{21}$$

Hence, written in terms of the quantities μ_n, ν_n and δ_n, we have

$$\begin{aligned}\|\gamma_l\| \cdot \|\nabla \tilde{S}_n(\gamma_l)\| \leq& 2(\mu_n\nu_n\delta_n + 3\mu_n\nu_n^2\delta_n + \mu_n\delta_n^2\nu_n^2) \\ =& 2\mu_n\nu_n\delta_n(1 + 3\nu_n + \delta\nu_n)\end{aligned} \tag{22}$$

□

Lemma A.5. *With* $\delta_n = (\mathbb{E}\|\Delta B_n\|^6)^{\frac{1}{6}}$, $\mathbb{E}|\lambda_1(B_n)|^6 \leq 64 + 63\delta_n^6$

Proof of Lemma A.5.

$$\begin{aligned}\mathbb{E}|\lambda_1(B_n)|^6 =& \mathbb{E}\|B_n\|^6 \\ =& \mathbb{E}\|I + \Delta B_n\|^6 \\ \leq& \mathbb{E}(\|I\| + \|\Delta B_n\|)^6 \\ =& \mathbb{E}(1 + \|\Delta B_n\|)^6\end{aligned} \tag{23}$$

Note that

$$gma \qquad \mathbb{E}(1+X)^6 = \mathbb{E}X^6 + 6\mathbb{E}X^5 + 15\mathbb{E}X^4 + 20\mathbb{E}X^3 + 15\mathbb{E}X^2 + 6\mathbb{E}X + 1 \tag{24}$$

and for $k \in \{1, 2, 3, 4, 5\}$, if X is a nonnegative random variable,

$$\begin{aligned}\mathbb{E}X^k =& \mathbb{1}_{X>1}\mathbb{E}X^k + \mathbb{1}_{X\leq 1}\mathbb{E}X^k \\ \leq& 1 + \mathbb{1}_{X\leq 1}\mathbb{E}X^6 \\ \leq& 1 + \mathbb{E}X^6\end{aligned} \tag{25}$$

Therefore, $\mathbb{E}|\lambda_1(B_n)|^6 \leq 64 + 63\mathbb{E}\|\Delta B_n\|^6$. □

From now on, for simplicity, we introduce $\delta_n' = (64 + 63\delta_n^6)^{\frac{1}{6}}$, as a function of δ_n.

Lemma A.6. $\forall \gamma \in \mathbb{R}^M$,

$$\begin{aligned}\|\gamma_l\|^2 \cdot \|\nabla^2 S_n(\gamma) - \nabla^2 \tilde{S}_n(\gamma)\| \leq & \mu_n \nu_n \delta_n (10 + 14\nu_n + 2\delta_n \nu_n + 16\nu_n^2 + 16\delta_n' \nu_n \\ & + 8\delta_n' \nu_n^2 + 8\delta_n' \nu_n + 8\delta_n \delta_n' \nu)\end{aligned} \tag{26}$$

Proof of Lemma A.6.

$$\nabla^2 S_n(\gamma) - \nabla^2 \tilde{S}_n(\gamma) = 2\mathbb{E}[H_1] - 2\mathbb{E}[H_2] + 8\mathbb{E}[H_2] - 8\mathbb{E}[H_4] \tag{27}$$

where

$$H_1 = \frac{(\gamma^T\gamma)A_n - (\gamma^T B_n \gamma)A_n}{(\gamma^T B_n \gamma)(\gamma^T \gamma)} \tag{28}$$

$$H_2 = \frac{(\gamma^T A_n \gamma)[(\gamma^T\gamma)^2 B_n - (\gamma^T B_n \gamma)^2]I)}{(\gamma^T B \gamma)^2 (\gamma^T \gamma)^2} \tag{29}$$

$$H_3 = \frac{(\gamma^T A_n \gamma)[(\gamma^T\gamma)^3 B_n \gamma\gamma^T B_n^T - (\gamma^T B_n \gamma)^3 \gamma\gamma^T]}{(\gamma^T B_n \gamma)^3 (\gamma^T \gamma)^3} \tag{30}$$

$$H_4 = \frac{(\gamma^T\gamma)^2 A_n \gamma\gamma^T B_n - (\gamma^T B_n \gamma)^2 A \gamma\gamma^T}{(\gamma^T B_n \gamma)^2 (\gamma^T \gamma)^2} \tag{31}$$

Thus, $\|\nabla^2 S_n(\gamma) - \nabla^2 \tilde{S}_n(\gamma)\| \leq 2\mathbb{E}\|H_1\| + 2\mathbb{E}\|H_2\| + 8\mathbb{E}\|H_3\| + 8\mathbb{E}\|H_4\|$, and we try to bound each term on the right hand side separately.

For the first term, there is

$$\|H_1\| \leq \frac{1}{\|\gamma\|^2} \frac{\|\Delta B_n\| |\lambda_1(A_n)|}{|\lambda_M(B_n)|} \tag{32}$$

Applying generalized Hölder's inequality, we obtain

$$\begin{aligned}\|\gamma\|^2 \cdot \mathbb{E}\|H_1\| \leq & \left(\mathbb{E}\frac{1}{|\lambda_M(B_n)|^3}\right)^{\frac{1}{3}} (\mathbb{E}|\lambda_1(A_n)|^3)^{\frac{1}{3}} (\mathbb{E}\|\Delta B_n\|^3)^{\frac{1}{3}} \\ \leq & \mu_n \nu_n \delta_n \ .\end{aligned} \tag{33}$$

For the second term, there is

$$H_2 = \frac{(\gamma^T A_n \gamma)[(\gamma^T\gamma)^2 \Delta B_n - 2(\gamma^T\gamma)(\gamma^T \Delta B_n \gamma)I - (\gamma^T \Delta B_n \gamma)^2 I]}{(\gamma^T B_n \gamma)^2 (\gamma^T \gamma)^2} \tag{34}$$

Hence,

$$\|H_2\| \leq \frac{1}{\|\gamma\|^2} \frac{1}{\lambda_M^2(B_n)} |\lambda_1(A_n)| (3\|\Delta B_n\| + \|\Delta B_n\|^2) \tag{35}$$

Applying generalized Hölder's inequality, we obtain

$$\begin{aligned}\|\gamma\|^2 \cdot \mathbb{E}\|H_2\| \leq & \left(\mathbb{E}\frac{3}{|\lambda_M(B_n)|^6}\right)^{\frac{1}{3}} (\mathbb{E}|\lambda_1(A_n)|^3)^{\frac{1}{3}} (\mathbb{E}\|\Delta B_n\|^3)^{\frac{1}{3}} \\ & + \left(\mathbb{E}\frac{3}{|\lambda_M(B_n)|^6}\right)^{\frac{1}{3}} (\mathbb{E}|\lambda_1(A_n)|^3)^{\frac{1}{3}} (\mathbb{E}\|\Delta B_n\|^6)^{\frac{1}{3}} \\ \leq & \mu_n \nu_n \delta_n (3\nu_n + \delta_n \nu_n)\end{aligned} \tag{36}$$

For H_3, note that

$$\begin{aligned}(\gamma^T\gamma)^3 B_n \gamma\gamma^T B_n^T - (\gamma^T B_n \gamma)^3 \gamma\gamma^T = & (\gamma^T\gamma)^3 (B_n - I)\gamma\gamma^T B_n + (\gamma^T\gamma)^3 \gamma\gamma^T (B_n - I) \\ & + [(\gamma^T\gamma)^3 - (\gamma^T B_n \gamma)^3]\gamma\gamma^T \\ = & (\gamma^T\gamma)^3 \Delta B_n \gamma\gamma^T B_n + (\gamma^T\gamma)^3 \gamma\gamma^T \Delta B_n \\ & + [(\gamma^T B_n \gamma)^2 (-\gamma^T \Delta B_n \gamma)\gamma\gamma^T + (\gamma^T B_n \gamma)(-\gamma^T \Delta B_n \gamma)\gamma\gamma^T \\ & + (-\gamma^T \Delta B_n \gamma)\gamma\gamma^T]\end{aligned} \tag{37}$$

Hence,

$$\begin{aligned} H_3 =&(\gamma^T A_n \gamma)\Big[\frac{(\gamma^T\gamma)^3 \Delta B_n \gamma\gamma^T B_n + (\gamma^T\gamma)^3\gamma\gamma^T\Delta B_n + (-\gamma^T\Delta B_n\gamma)\gamma\gamma^T}{(\gamma^T B_n\gamma)^3(\gamma^T\gamma)^3} \\ &+ \frac{(-\gamma^T\Delta B_n\gamma)\gamma\gamma^T}{(\gamma^T B_n\gamma)^2(\gamma^T\gamma)} + \frac{(-\gamma^T\Delta B_n\gamma)\gamma\gamma^T}{(\gamma^T B_n\gamma)(\gamma^T\gamma)^2}\Big] \end{aligned} \tag{38}$$

Thus,

$$\|H_3\| \leq \frac{|\lambda_1(A_n)|}{\|\gamma\|^2}\left[\frac{1}{|\lambda_M^3(B_n)|}(\|\Delta B_n\||\lambda_1(B_n)| + 2\|\Delta B_n\|) + \frac{1}{\lambda_M^2(B_n)}\|\Delta B_n\| + \frac{1}{|\lambda_M(B_n)|}\|\Delta B_n\|\right] \tag{39}$$

Applying generalized Hölder's inequality, we obtain

$$\begin{aligned} \|\gamma\|^2\cdot\mathbb{E}\|H_3\| \leq& \left(\mathbb{E}\frac{1}{|\lambda_M(B_n)|^6}\right)^{\frac{1}{2}}(\mathbb{E}|\lambda_1(A_n)|^6)^{\frac{1}{6}}(\mathbb{E}\|\Delta B_n\|^6)^{\frac{1}{6}}(\mathbb{E}|\lambda_1(B_n)|^6)^{\frac{1}{6}} \\ &+ 2\left(\mathbb{E}\frac{1}{|\lambda_M(B_n)|^6}\right)^{\frac{1}{2}}(\mathbb{E}|\lambda_1(A_n)|^3)^{\frac{1}{3}}(\mathbb{E}\|\Delta B_n\|^6)^{\frac{1}{6}} \\ &+ \left(\mathbb{E}\frac{1}{|\lambda_M(B_n)|^6}\right)^{\frac{1}{3}}(\mathbb{E}|\lambda_1(A_n)|^3)^{\frac{1}{3}}(\mathbb{E}\|\Delta B_n\|^3)^{\frac{1}{3}} \\ &+ \left(\mathbb{E}\frac{1}{|\lambda_M(B_n)|^3}\right)^{\frac{1}{3}}(\mathbb{E}|\lambda_1(A_n)|^3)^{\frac{1}{3}}(\mathbb{E}\|\Delta B_n\|^3)^{\frac{1}{3}} \\ \leq& \mu_n\nu_n\delta_n(\delta_n'\nu_n^2 + 2\nu_n^2 + \nu_n + 1) \end{aligned} \tag{40}$$

For the last term,

$$H_4 = \frac{[-2(\gamma^T\gamma)(\gamma^T\Delta B_n\gamma)I - (\gamma^T\Delta B_n\gamma)^2 I]A_n\gamma\gamma^T B_n + (\gamma^T B_n\gamma)^2 A_n\gamma\gamma^T\Delta B_n}{(\gamma^T B_n\gamma)^2(\gamma^T\gamma)^2} \tag{41}$$

Thus,

$$\|H_4\| \leq \frac{1}{\|\gamma\|^2}\left[\frac{1}{\lambda_M^2(B_n)}(2\|\Delta B_n\| + \|\Delta B_n\|^2)|\lambda_1(A_n)||\lambda_1(B_n)| + \frac{1}{\lambda_M^2(B_n)}|\lambda_1^2(B_n)||\lambda_1(A_n)|\|\Delta B_n\|\right] \tag{42}$$

Applying generalized Hölder's inequality, we obtain

$$\begin{aligned} \|\gamma\|^2\cdot\mathbb{E}\|H_4\| \leq& 2\left(\mathbb{E}\frac{1}{|\lambda_M(B_n)|^6}\right)^{\frac{1}{3}}(\mathbb{E}|\lambda_1(A_n)|^3)^{\frac{1}{3}}(\mathbb{E}\|\Delta B_n\|^6)^{\frac{1}{6}}(\mathbb{E}|\lambda_1(B_n)|^6)^{\frac{1}{6}} \\ &+ \left(\mathbb{E}\frac{1}{|\lambda_M(B_n)|^6}\right)^{\frac{1}{3}}(\mathbb{E}|\lambda_1(A_n)|^6)^{\frac{1}{6}}(\mathbb{E}\|\Delta B_n\|^6)^{\frac{1}{3}}(\mathbb{E}|\lambda_1(B_n)|^6)^{\frac{1}{6}} \\ &+ \left(\mathbb{E}\frac{1}{|\lambda_M(B_n)|^6}\right)^{\frac{1}{3}}(\mathbb{E}|\lambda_1(A_n)|^6)^{\frac{1}{6}}(\mathbb{E}\|\Delta B_n\|^6)^{\frac{1}{6}}(\mathbb{E}|\lambda_1(B_n)|^6)^{\frac{1}{3}} \\ \leq& \mu_n\nu_n\delta_n(2\nu_n\delta_n' + \delta_n\delta_n'\nu_n + {\delta_n'}^2\nu_n) \end{aligned} \tag{43}$$

Therefore, summing up the bounds above, we obtain

$$\begin{aligned} \|\gamma_l\|^2\cdot\|\nabla^2 S_n(\gamma) - \nabla^2\tilde{S}_n(\gamma)\| \leq& \mu_n\nu_n\delta_n(10 + 14\nu_n + 2\delta_n\nu_n + 16\nu_n^2 + 16\delta_n'\nu_n \\ &+ 8\delta_n'\nu_n^2 + 8\delta_n'\nu_n + 8\delta_n\delta_n'\nu) \end{aligned} \tag{44}$$

Hence, combining inequality 15, Lemma A.4 and Lemma A.6, we get

$$\begin{aligned} 1 - \cos^2(\gamma_l, \tilde{\gamma}_g) \leq& \eta_n[4\mu_n\nu_n\delta_n(1 + 3\nu_n\delta_n\mu_n) + \frac{1}{2}\mu_n\nu_n\delta_n(10 + 14\nu_n + 2\delta_n\nu_n + 16\nu_n^2 \\ &+ 16\delta_n'\nu_n + 8\delta_n'\nu_n^2 + 8\delta_n'\nu_n + 8\delta_n\delta_n'\nu)] \\ =& \mu_n\nu_n\delta_n\eta_n(9 + 19\nu_n + 5\delta_n\nu_n + 8\nu_n^2 + 8\delta_n'\nu_n + 4\delta_n'{\nu_n}^2 + 4\delta_n'\nu_n + 4\delta_n\delta_n'\nu_n) \end{aligned} \tag{45}$$

For simplicity, we define $C(\delta_n, \nu_n) = 9+19\nu_n+5\delta_n\nu_n+8\nu_n^2+8\delta_n'\nu_n+4\delta_n'{\nu_n}^2+4\delta_n'\nu_n+4\delta_n\delta_n'\nu_n$. Thus,

$$1 - \cos^2(\gamma_l, \tilde{\gamma}_g) \leq \mu_n\nu_n\delta_n\eta_n C(\delta_n, \nu_n) \tag{46}$$

□

Following the notations in the proof of Lemma A.3, we write $\gamma_l = \sum_{i=1}^M \alpha_i\hat{\gamma}_i$. Note that $\tilde{\gamma}_g = \pm\hat{\gamma}_1$, and $|\cos(\gamma, \hat{\gamma}_i)| = |\bar{\alpha}_i|$. Thus,

$$\begin{aligned}\tilde{L}_n(\beta_l) =& \tilde{S}_n(\gamma_l)\\ =& \frac{\sum_{i=1}^M \alpha_i^2 l_i}{\sum_{i=1}^M \alpha_i^2} = \sum_{i=1}^M \bar{\alpha}_i^2 l_i\end{aligned} \tag{47}$$

Since Y_n is positive semidefinite, $\mathbb{E}Y_n$ is also positive semidefinite, and hence $\bar{A}_n = R_n^T\mathbb{E}Y_n(R_n^{-1})^T$ is positive semidefinite as well. This means that $l_i \geq 0, \forall i \in \{1, ..., M\}$. Since $\tilde{L}_n(\tilde{\beta}_g) = \tilde{S}_n(\tilde{\gamma}_g) = \tilde{S}_n(\hat{\gamma}_1) = l_1$, there is

$$|\tilde{L}_n(\tilde{\beta}_g) - \tilde{L}_n(\beta_l)| \leq (1 - \bar{\alpha}_1^2)l_1 \leq (1 - \cos^2(\gamma_l, \tilde{\gamma}_g))\lambda_1(\bar{A}_n) \tag{48}$$

Next, we bound the first and the third term on the right hand side of the inequality in Lemma A.1.

Lemma A.7. $\forall\beta$,

$$|L_n(\beta) - \tilde{L}_n(\beta)| \leq (\mathbb{E}\|\Delta B_n\|^3)^{\frac{1}{3}} \cdot (\mathbb{E}|\lambda_1(A_n)|^3)^{\frac{1}{3}} \cdot \left(\mathbb{E}|\frac{1}{\lambda_M(B_n)}|^3\right)^{\frac{1}{3}} \tag{49}$$

Proof of Lemma A.7. Let $\gamma = T_n^T\beta$.

$$\begin{aligned}|L_n(\beta) - \tilde{L}_n(\beta)| =& S_n(\gamma) - \tilde{S}_n(\gamma)\\ =& \left|\mathbb{E}\frac{(\gamma^T\Delta B_n\gamma)(\gamma^T A_n\gamma)}{(\gamma^T B_n\gamma)(\gamma^T\gamma)}\right|\\ \leq& \mathbb{E}\frac{\|\Delta B_n\||\lambda_1(A_n)|}{|\lambda_M(B_n)|}\end{aligned} \tag{50}$$

Thus, we get the desired lemma by the generalized Hölder's inequality.

□

Combining inequality 46, inequality 48 and Lemma A.7, we get

$$\begin{aligned}|L_n(\beta_l) - L_n(\beta_g)| \leq& 2(\mathbb{E}\|\Delta B_n\|^3)^{\frac{1}{3}} \cdot (\mathbb{E}|\lambda_1(A_n)|^3)^{\frac{1}{3}} \cdot \left(\mathbb{E}\left|\frac{1}{\lambda_M(B_n)}\right|^3\right)^{\frac{1}{3}} + (1 - \cos^2(\gamma_l, \tilde{\gamma}_g))\lambda_1(\bar{A}_n)\\ \leq& 2\mu_n\nu_n\delta_n + \mu_n\nu_n\delta_n\eta_n C(\delta_n, \nu_n) \cdot \lambda_1(\bar{A}_n)\end{aligned} \tag{51}$$

Meanwhile,

$$\begin{aligned}|L_n(\beta_g) - \tilde{L}_n(\tilde{\beta}_g)| \leq& \max\{|L_n(\beta_g) - \tilde{L}_n(\beta_g)|, |L_n(\tilde{\beta}_g) - \tilde{L}_n(\tilde{\beta}_g)|\}\\ \leq& (\mathbb{E}\|\Delta B_n\|^3)^{\frac{1}{3}} \cdot (\mathbb{E}|\lambda_1(A_n)|^3)^{\frac{1}{3}} \cdot (\mathbb{E}|\frac{1}{\lambda_M(B_n)}|^3)^{\frac{1}{3}}\\ \leq& \mu_n\nu_n\delta_n\end{aligned} \tag{52}$$

Hence,

$$\begin{aligned}L_n(\beta_g) \geq& \tilde{L}_n(\tilde{\beta}_g) - \mu_n\nu_n\delta_n\\ \geq& \lambda_1(\bar{A}_n) - \mu_n\nu_n\delta_n\\ \geq& \eta_n^{-1} - \mu_n\nu_n\delta_n\end{aligned} \tag{53}$$

, or

$$\lambda_1(\bar{A}_n) \leq L_n(\beta_g) + \mu_n \nu_n \delta_n \tag{54}$$

Therefore,

$$\begin{aligned}
|L_n(\beta_l) - L_n(\beta_g)| \leq & 2\mu_n\nu_n\delta_n + (1-\cos^2(\gamma_l, \tilde{\gamma}_g))[L_n(\beta_g) + \mu_n\nu_n\delta_n] \\
\leq & \mu_n\nu_n\delta_n[2 + \eta_n\mu_n\nu_n\delta_n C(\delta_n,\nu_n)] + \eta_n\mu_n\nu_n\delta_n C(\delta_n,\nu_n) L_n(\beta_g) \\
\leq & L_n(\beta_g)\left\{\frac{\mu_n\nu_n\delta_n[2+\eta_n\mu_n\nu_n\delta_n C(\delta_n,\nu_n)]}{\eta_n^{-1} - \mu_n\nu_n\delta_n} + \eta_n\mu_n\nu_n\delta_n C(\delta_n,\nu_n)\right\} \\
= & \frac{2\eta_n\mu_n\nu_n\delta_n[2+C(\delta_n,\nu_n)]}{1-\eta_n\mu_n\nu_n\delta_n} L_n(\beta_g)
\end{aligned} \tag{55}$$

Hence, we have proved the theorem, with $\epsilon_{\eta_n,\mu_n,\nu_n,\delta_n} = \frac{2\eta_n\mu_n\nu_n\delta_n[2+C(\delta_n,\nu_n)]}{1-\eta_n\mu_n\nu_n\delta_n}$. □

B BACKGROUND

B.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.

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 (Newman, 2006). Note that $\sum_{i,j} A_{i,j} = s^T D s$, with $D = \text{diag}(A\mathbf{1})$ (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 (Newman, 2006) under appropriate conditions, and is associated with the graph conductance (Spielman, 2015) 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 $\tilde{A} = \|\mathcal{L}(A)\| - \mathcal{L}(A)$, and then iterating

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

Unrolling power iterations and recasting the resulting model as a trainable neural network is akin to the LISTA (Gregor & LeCun, 2010) sparse coding model, which unrolled iterative proximal splitting algorithms.

Despite the appeal of graph Laplacian spectral approaches, it is well known (Krzakala et al., 2013) 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.

B.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

$$\mathbb{P}(\sigma) = \frac{1}{\mathcal{Z}} \prod_{i \in V} \phi_i(\sigma_i) \prod_{ij \in E} \psi_{ij}(\sigma_i, \sigma_j). \tag{56}$$

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

$$b_{i \to j}(\sigma_i) = \frac{1}{Z_{i \to j}} \phi_i(\sigma_i) \prod_{k \in \delta i \setminus j} \sum_{\sigma_k \in X} \psi_{ik}(\sigma_i, \sigma_k) b_{k \to i}(\sigma_k). \tag{57}$$

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 graphs, like the Stochastic Blockmodel with constant degree (MNS) 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.

B.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); 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., 2014) 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) (Saade et al., 2014). 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.

B.4 STOCHASTIC BLOCK MODEL

We briefly review the main properties needed in our analysis, and refer the interested reader to (Abbe, 2017) for an excellent recent review. The Stochastic Blockmodel (SBM) is a random graph model denoted by $SBM(n, p, q, C)$. Implicitly there is an $F : V \to \{1, C\}$ 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 $\bar{F}_n : V \to \{1, C\}$ be our predicted community labels for $SBM(n, p, q, C)$, 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 $\frac{a+b}{2} \geq 1 + \sqrt{ab}$ (Mossel et al., 2014; Abbe et al., 2014). 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 (Abbe, 2017; Mossel et al., 2014)). 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 (Abbe, 2017) due to large fluctuations in the degree distribution that prevent eigenvectors form concentrating on the clusters. (Decelle et al., 2011) first proposed the BP algorithm on the SBM, which was proven to yield Bayesian optimal values in (Coja-Oghlan et al., 2016).

In the constant degree regime with balanced k communities, the Kesten-Stigum threshold is given by $SNR := (a - b)^2/(k(a + (k + 1)b))$ (Abbe, 2017). 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 algorithm. For $k > 4$ 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 $SNR > 1$ (Abbe, 2017; Decelle et al., 2011). The existence of the gap was shown by (Abbe, 2017) by proving a non-polynomial algorithm can do detection for some $SNR < 1$.

C GEOMETRIC BLOCK MODEL

Table 3: Overlap performance (in percentage) of the GNN model on the Geometric Block Model compared with several Spectral Approaches

Model	$S = 1$	$S = 2$	$S = 4$
Norm. Laplacian	1 ± 0.5	1 ± 0.6	1 ± 1
Bethe Hessian	18 ± 1	38 ± 1	38 ± 2
GNN G	20 ± 0.4	39 ± 0.5	39 ± 0.5
GNN $\{G, L(G)\}$	$\mathbf{22 \pm 0.4}$	$\mathbf{50 \pm 0.5}$	$\mathbf{76 \pm 0.5}$

The success of Belief-Propagation on the SBM relies on its locally hyperbolic properties, which make it tree-like with high probability. This behavior is completely different if one considers random graphs with locally Euclidean geometry. The Geometric Block Model (Sankararaman & Baccelli, 2018) is a random graph generated as follows. One starts by sampling n iid points $x_1, \ldots, x_n$ from a Gaussian mixture model given by means $\mu_1, \ldots \mu_k \in \mathbb{R}^d$ at distance S apart and identity covariances. The labels correspond to which Gaussian each sample belongs to. We draw an edge between two nodes i, j if $\|x_i - x_j\| \leq T/\sqrt{n}$. Due to the triangle inequality, the model contains a large number of short cycles, impacting the performance of loopy belief propagation. This motivates other estimation algorithms based on motif-counting (Sankararaman & Baccelli, 2018) that require knowledge of the model likelihood function. Table 3 shows the performance of the GNN models on the binary GBM model, obtained with $d = 2$, $n = 500$, $T = 5\sqrt{2}$ and varying S, and its comparison with several spectral methods, including the Bethe Hessian, which approximates BP around its stationary solution. We verify that the line GNN model, thanks to its added flexibility and the multiscale nature of its generators, is able to significantly outperform both spectral methods as well as the baseline GNN.

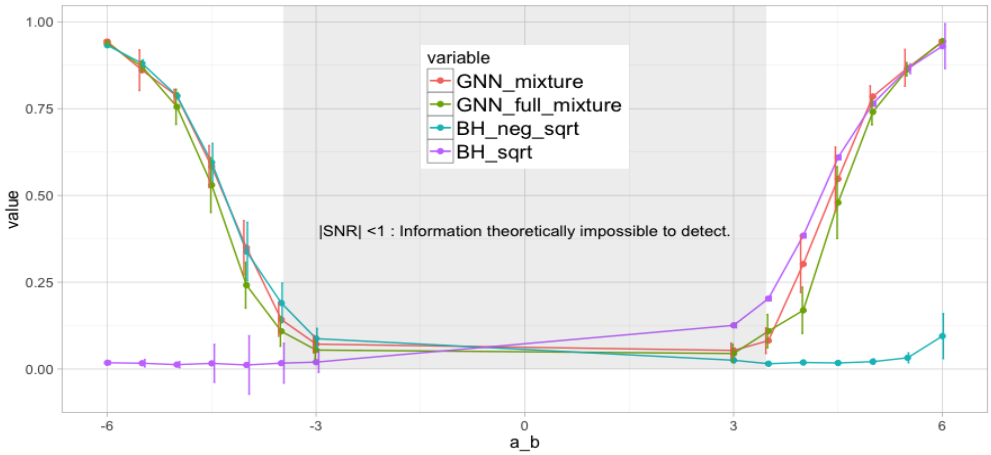

Figure 4. GNN mixture (Graph Neural Network trained on a mixture of SBM with average degree 3), GNN full mixture (GNN trained over different SNR regimes), $BH(\sqrt{\bar{d}})$ and $BH(-\sqrt{\bar{d}})$. *left:* $k = 2$. We verify that $BH(r)$ models cannot perform detection at both ends of the spectrum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

D FURTHER EXPERIMENTS ON THE SBM

We report here our experiments on the SBM mixture, generated with

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

where the average degree $\bar{d}$ is either fixed constant or also randomized with $\bar{d} \sim \text{Unif}(1, t)$. Figure 4 shows the overlap obtained by our model compared with several baselines. Our GNN model is either competitive with BH or beats BH, which achieves the state of the art along with BP (Saade et al., 2014), 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.