Optimal Performance of Graph Convolutional Networks on the Contextual Stochastic Block Model

Guillaume Dalle
Information and Network Dynamics (INDY) laboratory
Information, Learning and Physics (IdePHICS) laboratory
Statistical Physics of Computation (SPOC) laboratory
École polytechnique fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland
guillaume.dalle@epfl.ch

Patrick Thiran
Information and Network Dynamics (INDY) laboratory
École polytechnique fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland
patrick.thiran@epfl.ch

Abstract

For Graph Neural Networks, oversmoothing denotes the homogenization of vertex embeddings as the number of layers increases. To better understand this phenomenon, we study community detection with a linearized Graph Convolutional Network on the Contextual Stochastic Block Model. We express the distribution of the embeddings in each community as a Gaussian mixture over a low-dimensional latent space, with explicit formulas in the case of a single layer. This yields tractable estimators for classification accuracy at finite depth. Numerical experiments suggest that modeling with a single Gaussian is insufficient and that the impact of depth may be more complex than previously anticipated.

1 Introduction

1.1 Motivation

Graph Neural Networks (GNNs) are a family of neural networks designed to process relational data [8, 21, 38]. Despite their empirical successes, they suffer from a few shortcomings which hinder the use of deep networks. Most notably, the *oversmoothing* phenomenon [37] forces all vertex embeddings to converge to the same value as the number of layers goes to infinity. This makes the *finite-depth* regime very relevant since traditional GNNs can only deliver good performance before oversmoothing occurs. In this setting, recent works [27, 42, 44] offer new results on the classification accuracy of GNNs. Our goal is to extend and refine their insights by explicitly modeling the distribution of embeddings as a Gaussian mixture (see Figures 1 and 5). One can also try to mitigate oversmoothing with better models [15], but such is not our focus here.

1.2 Related work

Among GNN architectures, the most widely used is the Graph Convolutional Network (GCN) introduced by Kipf and Welling [28]. To facilitate analysis, the GCN is sometimes simplified as a sequence of fixed graph convolutions (without parameters or nonlinear activations), followed by a learnable final classifier. Proponents of this *linearized GCN* argue that it exhibits the same qualitative behavior as its nonlinear counterpart [43]. Indeed, at least for homophilous datasets (where nodes are

G. Dalle and P. Thiran, Optimal Performance of Graph Convolutional Networks on the Contextual Stochastic Block Model . *Proceedings of the Third Learning on Graphs Conference (LoG 2024)*, PMLR 269, Virtual Event, November 26–29, 2024.

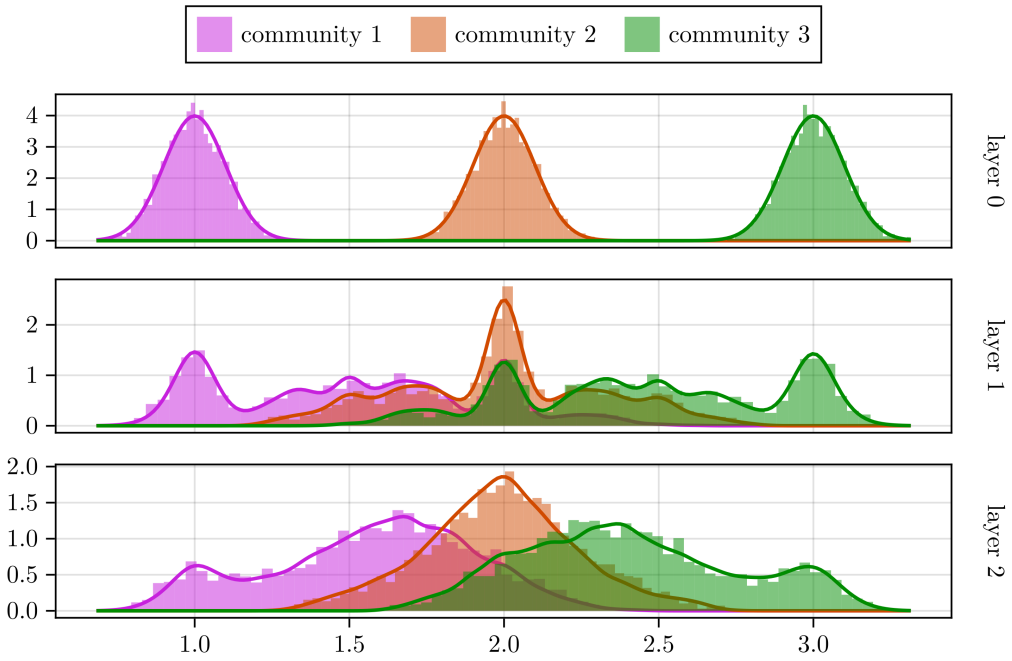

Figure 1: Embedding distributions in 1D are Gaussian mixtures.
Histograms are empirical samples, lines represent our density estimates.
Parameters: $N = 100$, $C = 3$, $p_{\text{in}} = 0.03$, $p_{\text{out}} = 0.02$, $\sigma = 0.1$. Estimation: $G = 100$.

more likely to connect with other nodes of the same community), the key ingredient is the low-pass filter induced by neighborhood averaging [34].

The Stochastic Block Model (SBM) [23] is a cornerstone of graph machine learning as it concisely represents sets of entities whose behavior only depends on the group they belong to (see Abbe [1] for a detailed review). It generates random graphs in which the vertices are split into communities. The existence likelihood of an edge only depends on the communities of both its endpoints. Community assignments are most often latent variables which must be recovered (either partially or completely) from observation of the graph structure. This community detection task is a common benchmark for GNN architectures, an early example being the article by Bruna and Li [6]. In the *Contextual SBM* (CSBM) introduced by Deshpande et al. [14], nodes also receive attributes in the form of multivariate Gaussian features. Within a community, vertex features are independent and identically distributed, matching the intuition that communities are homogeneous. The recovery threshold for community detection in the CSBM was first conjectured by Deshpande et al. [14] and then proven by Lu and Sen [31]. It depends on both the strength of the graph signal (how edge probabilities differ between communities) and the strength of the feature signal (how Gaussian distributions differ between communities). Optimal recovery algorithms for the CSBM typically involve Bayesian approximate message passing [4, 18]. This explains why message-passing GNNs [41] are often applied to this model, although Duranthon and Zdeborová [17, 18] show that they remain suboptimal.

Oversmoothing is often presented as a straightforward consequence of spectral graph theory. When the number of layers goes to infinity, GCN vertex embeddings become uninformative exponentially fast [7, 9, 30, 35]. Depending on the aggregation mechanism, the vertex embeddings may not all converge to the same vector, but in general they become proportional to one another, a phenomenon described as *rank collapse* [36]. Yet, as Keriven [27] points out, the first few layers can still be useful if one smoothes "not too little, not too much". In the right parameter regime, convolutions have a welcome denoising effect before the undesirable homogenization kicks in. To study these finite-depth phenomena, the CSBM has emerged as a standard setting and linearized GCNs as a prototypical architecture on which to prove analytical bounds. The statistical physics literature is rich in precise results for one or a few convolutional layers [2, 3, 17, 39]. The contributions of Wu et al. [44]

and Wang, Baranwal, and Fountoulakis [42] are most closely related to our own work. They improve upon the qualitative theorems of Keriven [27] and obtain quantitative estimates for the optimal depth of a linearized GCN on a 2-community CSBM. Their main conclusion is that the optimal depth increases logarithmically with the graph size. With the standard convolution mechanism, performance degrades after that point [44], but corrected convolutions allow performance to plateau instead [42].

However, many of the references we just listed make hypotheses of asymptotic nature, where the graph size, the feature dimension, and occasionally the average node degree, are all taken to be large. Even when their conclusions hold in non-asymptotic situations, they often contain unspecified "large enough" constants or order-of-approximation bounds (expressed as $O(\dots)$ or $\Omega(\dots)$). In contrast, our approach aims to provide *error estimators with minimal hypotheses that can be computed numerically*. Extracting analytical bounds from these estimators is left for future investigations.

1.3 Contributions and outline

The present work seeks to quantify the performance of a linearized GCN on the CSBM. Accuracy is a function of how well the final classifier can separate the embeddings of vertices belonging to different communities. We complement existing results by focusing on the finite case, where all dimension parameters are fixed. In this important setting, we show that the relevant embedding distributions are not mere Gaussians, but *mixtures of Gaussians*. This fact is illustrated on Figures 1 and 5, where we clearly observe deviations from a unimodal normal density. Our explicit handling of Gaussian mixtures for node embeddings is reminiscent of the work of Errica [20], but they focused on nearest-neighbor graphs built from the feature vectors, whereas the edge structure of the CSBM we study is only influenced by the community assignments. To describe the mixtures, we prove an exact formula for one layer, as well as a low-dimensional integral for several layers. The Bayes optimal classifier is not explicit, but we propose numerically tractable Monte-Carlo estimates that work well for shallow GCNs in low-dimensional feature spaces. Our reasoning generalizes to any CSBM, even when the number of communities exceeds 2 and/or when the communities are unbalanced.

Before the mathematical arguments, let us try to convey the main intuition. Consider the case of a sum convolution on a graph with a single community (Erdős-Renyi): after one layer, the embedding of a vertex is the sum of the feature vectors of all its neighbors, each of which is normally distributed with mean $\boldsymbol{\mu}$. Therefore, the conditional expectation of the embedding given the graph depends on the number of neighbors. But the final classifier is not tailored to a particular graph: it must be optimal for the entire family of graphs sampled from our generative model. When we integrate over all possible graphs from that model, we find that the embedding distribution has one mode $d\boldsymbol{\mu}$ corresponding to each integer degree d present in the node degree distribution. The weight of mode $d\boldsymbol{\mu}$ is the likelihood of the associated degree d, in this case a Binomial distribution. As a result, the final classifier must be able to separate multimodal embedding distributions, given by mixtures of Gaussians.

We first describe the precise setting of our analysis in Section 2. We discuss our theoretical results in Section 3, while delaying the proofs to Appendix A. We present numerical experiments in Section 4, with more details in Appendix B, before concluding in Section 5.

2 Setting

2.1 Model and architecture

Let us consider random undirected graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of size $N = |\mathcal{V}|$, generated by an SBM with C communities. We always refer to these communities with their integer index c, using shortcuts such as "$v \in c$" to signify that vertex v belongs to community c. Each community $c \in \{1, \dots, C\}$ has a size $N_c := f_c N$, which accounts for a fixed fraction f_c of the total graph size. We take community assignments to be fixed (planted): the first N_1 vertices always belong to community 1, the following N_2 to community 2, and so on. The integer $N_{c_1,c_2} := N_{c_2} - \mathbf{1}\{c_1 = c_2\}$ is the number of possible neighbors of a vertex $u \in c_1$ inside community c_2. A pair of vertices $u \in c_1$ and $v \in c_2$ is connected by an edge with probability q_{c_1,c_2}. Moreover, each vertex $u \in c$ is endowed with a vector of features $\boldsymbol{x}_u$ of dimension F that follows a multivariate Gaussian $\mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$, where the parameters are constant in each community.

We denote by $\boldsymbol{A} \in \{0,1\}^{N \times N}$ the adjacency matrix of such a graph (which we abusively call "graph" as well), by $\boldsymbol{D} \in \mathbb{N}^{N \times N} := \operatorname{diag}(d_1, \dots, d_N)$ its diagonal degree matrix, and by $d_{u,c} :=$

$\sum_{v\in c} A_{u,v}$ the "community degree" of vertex u in community c. Their augmented counterparts (which include self-loops) are $\widetilde{\boldsymbol{A}} := \boldsymbol{A} + \boldsymbol{I}$, $\widetilde{\boldsymbol{D}} := \boldsymbol{D} + \boldsymbol{I}$ and $\widetilde{d}_{u,c} := d_{u,c} + \mathbf{1}\{u \in c\}$. We denote by $\boldsymbol{X} := (\boldsymbol{x}_1, \dots, \boldsymbol{x}_N)^\top \in \mathbb{R}^{N\times F}$ the features matrix, by $\boldsymbol{H} := (\boldsymbol{h}_1, \dots, \boldsymbol{h}_N)^\top \in \mathbb{R}^{N\times F}$ the embeddings matrix and by $\boldsymbol{y} := (y_1, \dots, y_N)^\top \in \{1, \dots, C\}^N$ the community assignment vector, which is our classification target. When we manipulate probabilities, we always consider both sources of randomness $\boldsymbol{A}$ and $\boldsymbol{X}$, and we shorten $\mathbb{P} := \mathbb{P}_{\boldsymbol{A},\boldsymbol{X}}$. We denote by $\mathcal{B}(p)$ the Bernoulli distribution, by $\mathcal{B}(n,p)$ the binomial distribution and by $\mathcal{P}(\lambda)$ the Poisson distribution. The notation $\mathcal{B}(n,p)[d]$ refers to the binomial density $\mathcal{B}(n,p)$ evaluated at point d, that is, $\mathcal{B}(n,p)[d] = \binom{n}{d} p^d (1-p)^{n-d}$.

To predict communities from the data $(\boldsymbol{A}, \boldsymbol{X})$, we use a linearized GCN with L convolutional layers including residual connections and mean aggregation, followed by a final classifier φ_θ with parameters θ (for instance a multi-layer perceptron). Formally, the network outputs $\widehat{\boldsymbol{y}} = \varphi_\theta(\boldsymbol{H})$ where the embeddings are

$$\boldsymbol{H} := \left(\widetilde{\boldsymbol{D}}^{-1}\widetilde{\boldsymbol{A}}\right)^L \boldsymbol{X}. \tag{1}$$

Given a vertex u, if we fix the graph $\boldsymbol{A}$, the embedding $\boldsymbol{h}_u$ is a linear function of $\boldsymbol{X}$. Hence, the conditional distribution $\mathbb{P}(\boldsymbol{h}_u \mid \boldsymbol{A})$ is Gaussian, an observation which is key in the proofs.

2.2 Bayes accuracy

Given enough training data and expressive power, the final classifier φ_θ can be trained to approximate the optimal Bayes classifier $\varphi^\star$: for an individual embedding $\boldsymbol{h}$,

$$\varphi^\star(\boldsymbol{h}) = \operatorname*{argmax}_c \mathbb{P}(y = c \mid \boldsymbol{h}) := \operatorname*{argmax}_c f_c\, \mathbb{P}(\boldsymbol{h} \mid y = c). \tag{2}$$

Accordingly, we need to compute the marginal embedding density in community c after L layers, considering both sources of randomness $\boldsymbol{A}$ and $\boldsymbol{X}$. We denote it by

$$\pi_c(\boldsymbol{h}) := \mathbb{P}(\boldsymbol{h} \mid y = c). \tag{3}$$

Once we have a functional form for π_c, we can deduce the Bayes accuracy $a := \mathbb{P}(\widehat{y} = y) \in [0,1]$ using either of the following integrals (see Appendix A.1 for the proof):

$$a = \int_{\mathbb{R}^F} \max_c f_c \pi_c = \int_{\mathbb{R}^F} \frac{\max_c f_c \pi_c}{\sum_c f_c \pi_c} \sum_c f_c \pi_c. \tag{4}$$

These two equivalent expressions correspond to different estimation methods. When the feature dimension F is small, we can leverage numerical quadrature of the integrand $\boldsymbol{h} \mapsto \max_c f_c \pi_c(\boldsymbol{h})$ (left-hand side). When F is large, Monte-Carlo approximation becomes the method of choice: draw K samples $\boldsymbol{h}^{(k)} \sim \sum_c f_c \pi_c$ and return the average $\frac{1}{K}\sum_k \frac{\max_c f_c \pi_c(\boldsymbol{h}^{(k)})}{\sum_c f_c \pi_c(\boldsymbol{h}^{(k)})}$ (right-hand side). The reformulation from left- to right-hand side is necessary for Monte-Carlo because we cannot sample from the unnormalized Lebesgue measure on $\mathbb{R}^F$.

Remark 1. *Accuracy can also be estimated by choosing an architecture for the classifier φ_θ, training it on random datasets and comparing its predictions to the true community assignments. This approach has two main drawbacks, which our method overcomes. First, it only evaluates the performance of a specific architecture, which may not be Bayes-optimal. Second, it relies on classifier training and hyperparameter tuning, which is computationally expensive and not guaranteed to yield the optimal θ.*

Remark 2. *Some authors study other metrics than accuracy to quantify divergence between community embedding distributions. For instance, Errica [20] leverages an explicit formula for the squared error distance between Gaussian mixtures to understand the impact of convolutions on a nearest-neighbor graph. Such an approach would also work here.*

3 Error estimation

In this section, we fix a vertex u belonging to community c. Because vertices within a community are interchangeable, the embedding distribution $\mathbb{P}(\boldsymbol{h}_u)$ does not depend on the choice of $u \in c$, for it is always equal to π_c. This invariance in u is enabled by the permutation-invariant architecture of the linearized GCN, and the fact that we integrate over all graphs $\boldsymbol{A}$ when computing the probability distribution $\mathbb{P}$.

3.1 Embeddings as Gaussian mixtures

We start by eliciting the connection between Gaussian mixtures and embedding distributions. Our central insight is that the number of mixture components is much smaller than the number of possible graphs $\boldsymbol{A}$, which scales as $\Theta(N^2)$. In fact, we prove that the information given by the graph about the embedding $\boldsymbol{h}_u$ can be *compressed into* $\Theta(C)$ *degrees of freedom*.

Theorem 1. *The embedding distribution π_c after one layer is a mixture of Gaussians with one component per possible value of the community degree vector $\boldsymbol{d}_u \in \{0,\dots,N_{c,1}\}\times\cdots\times\{0,\dots,N_{c,C}\}$:*

$$\pi_c = \sum_{\boldsymbol{d}_u} \mathbb{P}(\boldsymbol{d}_u)\,\mathcal{N}\big(\mathbb{E}[\boldsymbol{h}_u \mid \boldsymbol{d}_u], \mathbb{V}[\boldsymbol{h}_u \mid \boldsymbol{d}_u]\big). \tag{5}$$

The mixture moments are positive combinations of the original community parameters

$$\mathbb{E}[\boldsymbol{h}_u \mid \boldsymbol{d}_u] = \sum_{c_1} \frac{\widetilde{d}_{u,c_1}}{\widetilde{d}_u}\boldsymbol{\mu}_{c_1} \qquad \textit{and} \qquad \mathbb{V}[\boldsymbol{h}_u \mid \boldsymbol{d}_u] = \sum_{c_1} \frac{\widetilde{d}_{u,c_1}}{\widetilde{d}_u^2}\boldsymbol{\Sigma}_{c_1}, \tag{6}$$

while the mixture weights are products of Binomial densities evaluated at the community degrees

$$\mathbb{P}(\boldsymbol{d}_u) = \prod_{c_1=1}^{C} \mathcal{B}(N_{c,c_1}, q_{c,c_1})[d_{u,c_1}]. \tag{7}$$

Proof. See Appendix A.2. □

There are exactly $(N_{c,1}+1)(N_{c,2}+1)\cdots N_{c,c}\cdots(N_{c,C}+1)$ possible values for $\boldsymbol{d}_u$, a number of possibilities which scales as N^C.

Example 1. *Consider the case of $C=2$ communities with sizes $N_1 = N_2 = N/2$, connectivities $q_{1,1} = q_{2,2} = p_{\text{in}}$ and $q_{1,2} = q_{2,1} = p_{\text{out}}$, feature means $\boldsymbol{\mu}_1 = -\boldsymbol{\mu}_2 = \boldsymbol{\mu}$, and feature covariances $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \boldsymbol{\Sigma}$. We approximate $N_{1,2} = N_{2,1} = N/2 - 1 \approx N/2$ to shorten the formulas. Then the embedding distributions after one layer are:*

$$\pi_1 = \sum_{d_{11}=0}^{N/2}\sum_{d_{12}=0}^{N/2} \underbrace{\mathcal{B}\big(\tfrac{N}{2}, p_{\text{in}}\big)[d_{11}]\;\mathcal{B}\big(\tfrac{N}{2}, p_{\text{out}}\big)[d_{12}]}_{\textit{mixture weight}}\;\underbrace{\mathcal{N}\left(\frac{(d_{11}+1)-d_{12}}{(d_{11}+1)+d_{12}}\boldsymbol{\mu}, \frac{(d_{11}+1)+d_{12}}{((d_{11}+1)+d_{12})^2}\boldsymbol{\Sigma}\right)}_{\textit{mixture component}} \tag{8}$$

$$\pi_2 = \sum_{d_{21}=0}^{N/2}\sum_{d_{22}=0}^{N/2} \mathcal{B}\big(\tfrac{N}{2}, p_{\text{out}}\big)[d_{21}]\;\mathcal{B}\big(\tfrac{N}{2}, p_{\text{in}}\big)[d_{22}]\;\mathcal{N}\left(\frac{d_{21}-(d_{22}+1)}{d_{21}+(d_{22}+1)}\boldsymbol{\mu}, \frac{d_{21}+(d_{22}+1)}{(d_{21}+(d_{22}+1))^2}\boldsymbol{\Sigma}\right). \tag{9}$$

We now move on to the multi-layer case, where the mixture weights are no longer expressed explicitly.

Theorem 2. *Let $L \geq 1$. We define*

$$\widetilde{\boldsymbol{R}} := (\widetilde{\boldsymbol{D}}^{-1}\widetilde{\boldsymbol{A}})^L, \qquad \widetilde{s}_{u,c_1} := \sum_{v\in c_1}\widetilde{R}_{u,v}, \qquad \widetilde{t}_{u,c_1} := \sum_{v\in c_1}\widetilde{R}_{u,v}^2. \tag{10}$$

The embedding distribution π_c after L layers is a mixture of Gaussians with one component per possible value of the couple of vectors $(\widetilde{\boldsymbol{s}}_u, \widetilde{\boldsymbol{t}}_u)$ (which lives in $[0,1]^C \times [0,1]^C$):

$$\pi_c = \sum_{\widetilde{\boldsymbol{s}}_u, \widetilde{\boldsymbol{t}}_u} \mathbb{P}(\widetilde{\boldsymbol{s}}_u, \widetilde{\boldsymbol{t}}_u)\,\mathcal{N}\big(\mathbb{E}[\boldsymbol{h}_u \mid \widetilde{\boldsymbol{s}}_u, \widetilde{\boldsymbol{t}}_u], \mathbb{V}[\boldsymbol{h}_u \mid \widetilde{\boldsymbol{s}}_u, \widetilde{\boldsymbol{t}}_u]\big) \tag{11}$$

The mixture moments are positive combinations of the original community parameters

$$\mathbb{E}[\boldsymbol{h}_u \mid \widetilde{\boldsymbol{s}}_u, \widetilde{\boldsymbol{t}}_u] = \sum_{c_1} \widetilde{s}_{u,c_1}\boldsymbol{\mu}_{c_1} \qquad \textit{and} \qquad \mathbb{V}[\boldsymbol{h}_u \mid \widetilde{\boldsymbol{s}}_u, \widetilde{\boldsymbol{t}}_u] = \sum_{c_1} \widetilde{t}_{u,c_1}\boldsymbol{\Sigma}_{c_1}. \tag{12}$$

Proof. See Appendix A.3. □

The quantities $\boldsymbol{d}_u$ (Theorem 1), $\widetilde{\boldsymbol{s}}_u$ and $\widetilde{\boldsymbol{t}}_u$ (Theorem 2) can be thought of as some sort of "sufficient statistics", encoding the influence of the graph $\boldsymbol{A}$ on the embedding $\boldsymbol{h}_u$. Crucially, their distribution does not depend on the specific choice of the vertex u in the community c. Indeed, due to the symmetry properties of the CSBM, they have the same distribution for any other member u' of the same community.

Remark 3. *To better understand Equation (10), we recall that the convolution $(\boldsymbol{D}+\boldsymbol{I})^{-1}(\boldsymbol{A}+\boldsymbol{I})$ defines a random walk on the graph $\boldsymbol{A}$ (it is a stochastic matrix, whose rows are nonegative and sum to one). Thus, the coefficient $\widetilde{R}_{u,v} = [(\boldsymbol{D}+\boldsymbol{I})^{-1}(\boldsymbol{A}+\boldsymbol{I})]_{u,v}$ gives the probability for such a random walk, starting at u, to arrive at v after L steps. Note that this probability is itself a random variable, for it is a function of $\boldsymbol{A}$. With that in mind, $\widetilde{s}_{u,c_1}$ is the probability that a* single *random walk of length L, starting at u, arrives at some vertex v of community c_1. Similarly, $\widetilde{t}_{u,c_1}$ is the probability that* two *independent random walks of length L, both starting at u, both arrive at the* same *vertex v of community c_1.*

3.2 Tractable error estimation

We now turn Theorems 1 and 2 into *numerical* estimation procedures for the Bayes accuracy. There are two computationally challenging aspects: the integral over $\mathbb{R}^F$ in Equation (4), which we already discussed in Section 2.2, and the large number of mixture components in Equations (5) and (11).

For $L = 1$, the explicit mixture weights of Equation (7) reveal that *most mixture components bring a negligible contribution*. Indeed, in the typical case of a sparse SBM, the connectivity q_{c,c_1} scales as $1/N$, which means that the expected degree remains constant as the graph grows. Equivalently, the product $N_{c,c_1} q_{c,c_1}$ converges to a finite limit $\lambda_{c,c_1} > 0$ (interpreted as the average number of neighbors of a vertex $u \in c$ that belong to c_1). In Equation (7), we can then replace the Binomial weights with Poisson weights $\mathcal{P}(\lambda_{c,c_1})[d_{u,c_1}]$. By Bennet's inequality [40], as soon as $d_{u,c_1} \geq \alpha \lambda_{c,c_1}$ for some α, these Poisson weights decay exponentially with α. Therefore, a simple heuristic yielding a tractable number of components is to define a small threshold $\varepsilon > 0$ and approximate Equation (5) with

$$\widehat{\pi}_c = \sum_{\boldsymbol{d}_u : \mathbb{P}(\boldsymbol{d}_u) > \varepsilon} \mathbb{P}(\boldsymbol{d}_u)\, \mathcal{N}\big(\mathbb{E}[\boldsymbol{h}_u \mid \boldsymbol{d}_u], \mathbb{V}[\boldsymbol{h}_u \mid \boldsymbol{d}_u]\big) \tag{13}$$

Beyond one layer, the mixture weights $\mathbb{P}(\widetilde{\boldsymbol{s}}_u, \widetilde{\boldsymbol{t}}_u)$ are harder to compute because they originate from random walks on random graphs. As a workaround, we suggest drawing K Monte-Carlo samples $(\widetilde{\boldsymbol{s}}_u^{(k)}, \widetilde{\boldsymbol{t}}_u^{(k)})$ and approximating Equation (11) with

$$\widehat{\pi}_c = \frac{1}{K} \sum_{k=1}^{K} \mathcal{N}\big(\mathbb{E}[\boldsymbol{h}_u \mid \widetilde{\boldsymbol{s}}_u^{(k)}, \widetilde{\boldsymbol{t}}_u^{(k)}], \mathbb{V}[\boldsymbol{h}_u \mid \widetilde{\boldsymbol{s}}_u^{(k)}, \widetilde{\boldsymbol{t}}_u^{(k)}]\big) \tag{14}$$

By Equation (10), and since vertices within a community are interchangeable, generating a single graph $\boldsymbol{A}$ gives rise to N_c identically distributed (albeit not independent) samples with the same distribution as $(\widetilde{\boldsymbol{s}}_u, \widetilde{\boldsymbol{t}}_u)$.

4 Numerical experiments

We now present numerical experiments highlighting three main phenomena:

- The embedding mixture cannot be reduced to a single Gaussian, except in some limiting cases.
- Thanks to Theorem 1, we can trace frontiers between regions where convolution increases performance and regions where convolution hurts performance.
- Thanks to Theorem 2, we can plot performance curves as a function of depth that exhibit several local extrema.

4.1 Protocol

Our code is written in the Julia programming language [5] and available as a GitHub repository with precise reproduction instructions [12]. To simplify exposition, we present experiments where the communities have equal sizes $N_1 = \cdots = N_C$ and only two possible values for connectivities: q_{c_1,c_2} is equal to p_{in} if $c_1 = c_2$ and to p_{out} if $c_1 \neq c_2$. Note that the underlying code is completely generic and does not need such assumptions. We define two low-dimensional models:

Straight 1D (Figure 1) where the community means are scattered along the real line $\boldsymbol{\mu}_c = c$, at a distance 1 from their closest neighbor, and the community variances are $\boldsymbol{\Sigma}_c = \sigma^2$.

Circular 2D (Figure 5) where the community means are scattered along a circle $\boldsymbol{\mu}_c \propto (\cos(2\pi c/C), \sin(2\pi c/C))$ and the community covariance matrices are isotropic $\boldsymbol{\Sigma}_c = \mathrm{diag}(\sigma^2, \sigma^2)$. The circle radius is chosen so that community means are at distance 1 from their closest neighbor.

These models can be fully described with 5 parameters $(N, C, p_{\mathrm{in}}, p_{\mathrm{out}}, \sigma)$: the graph size N, the number of communities C, the probability p_{in} of being connected to a vertex from the same community, the probability p_{out} of being connected to a vertex from another community, the standard deviation of the features σ (expressed as a fraction of the smallest distance between two community means, because fixing the means and varying σ is the same as fixing σ and varying the means). All our experiments involve drawing graphs with their features from a CSBM described by one of these two models. Additional details are given in Appendix B.

4.2 Mixture versus single Gaussian

While Figures 1 and 5 visually support the use a mixture model, it is specific to a given set of parameters. Are there some parameter regimes that are well described by a single Gaussian, for which the mixture approach only brings unneeded complexity? To answer this question, we focus on the Straight 1D test case with $L = 1$ layer and $C = 2$ communities. We compute the embedding mixture for the first community π_1 and compare it with the single Gaussian whose moments are identical to those of the mixture, namely $\nu_1 := \mathcal{N}(\mathbb{E}[\pi_1], \mathbb{V}[\pi_1])$. Our metric is the total variation distance $\mathrm{TV}(\pi_1, \nu_1) = \frac{1}{2}\int |\pi_1(\boldsymbol{h}) - \nu_1(\boldsymbol{h})| \, \mathrm{d}\boldsymbol{h}$, which is 0 if the two densities are equal almost everywhere and 1 if they are as far apart as can be. The higher the total variation distance, the more we benefit from computing a full mixture.

On Figure 2, we show the impact of two parameters on the total variation distance: the feature standard deviation σ and the graph size N. Since we keep the connectivities (edge probabilities) p_{in} and p_{out} fixed, varying the graph size N amounts to varying the average number of neighbors. We observe that $\mathrm{TV}(\pi_1, \nu_1)$ is closer to 1 when σ is small and N is small. Conversely, $\mathrm{TV}(\pi_1, \nu_1)$ decreases as σ and N become large. Intuitively, when σ is large, the components in the Gaussian mixture are no longer well-separated, which makes unimodal approximation more reasonable.

4.3 Convolution benefits for one layer

We continue studying the Straight 1D test case with $L = 1$ layer and $C = 2$ communities. Our next question is whether graph convolution enhances or worsens classification performance. This is the same question asked by Keriven [27], except that his answer was of theoretical and approximate nature, while ours is numerical but exact.

In Figure 3, we fix the noise σ and let the connectivities p_{in} and p_{out} vary. The colors represent the impact of one graph convolution on classification accuracy: green if accuracy increases from $L = 0$ to $L = 1$, red if it decreases. We notice not one but two white lines, which correspond to homophilous and heterophilous graphs respectively. When $p_{\mathrm{in}} \gg p_{\mathrm{out}}$ or when $p_{\mathrm{out}} \gg p_{\mathrm{in}}$, one layer of graph convolution starts having a beneficial effect on community detection, while this same layer is detrimental when $p_{\mathrm{in}} \approx p_{\mathrm{out}}$. This echoes the observation by Ma et al. [32] that there are some forms of heterophily which still allow convolutions to improve performance. Interestingly, our one-layer estimation procedure works for any level of homophily in the graph. We additionally note that the roles of p_{in} and p_{out} are not symmetric, part of which is explained by the residual connection.

4.4 Performance curves for several layers

For our final experiment, we move to the Circular 2D test case and explore $C \geq 2$ communities. The goal of Figure 4 is to demonstrate the possibility of "zig-zagging" performance curves, when plotted as a function of depth. Indeed, the curves for $C \geq 4$ suggests that accuracy can first decrease, then increase, then decrease again. Therefore, focusing on the first layer alone is not enough to predict whether some number of convolutions can be helpful. As Keriven [27] detects beneficial smoothing based solely on improvements at depth 1, we conclude that his analysis can be too pessimistic when the first improvement only occurs later. We also note that increasing the number of communities

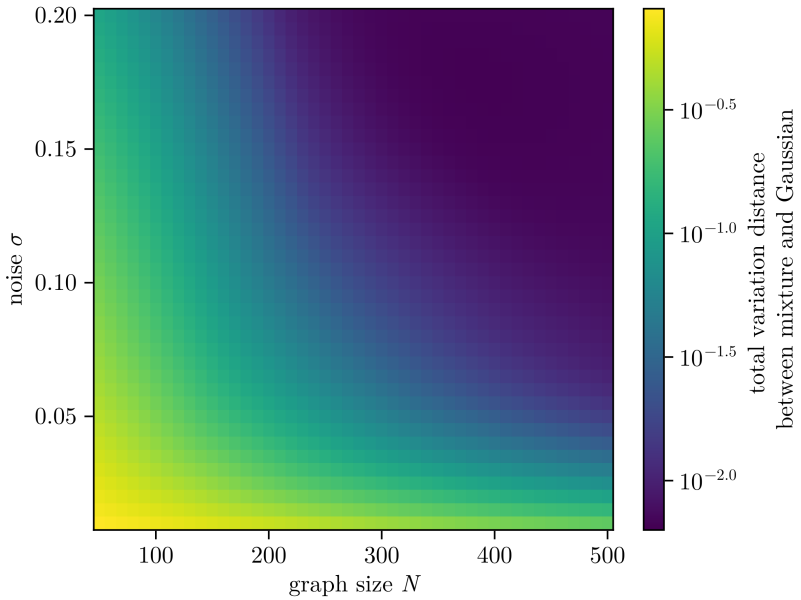

Figure 2: The embedding mixture degenerates into a single Gaussian for large noise or high degrees.
Parameters: $C = 2$, $p_{\text{in}} = 0.03$, $p_{\text{out}} = 0.02$

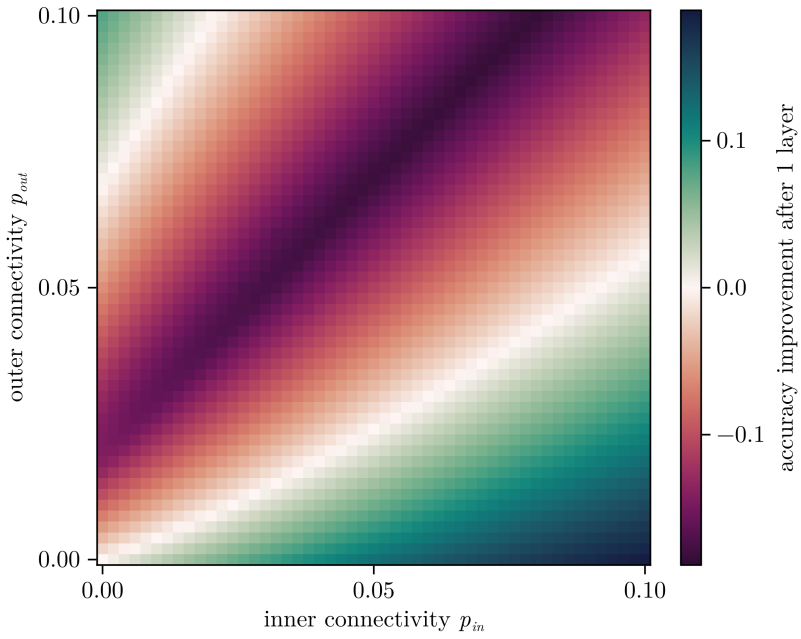

Figure 3: We can detect frontiers for the impact of one graph convolution on classification accuracy.
Parameters: $N = 100$, $C = 2$, $\sigma = 0.1$.

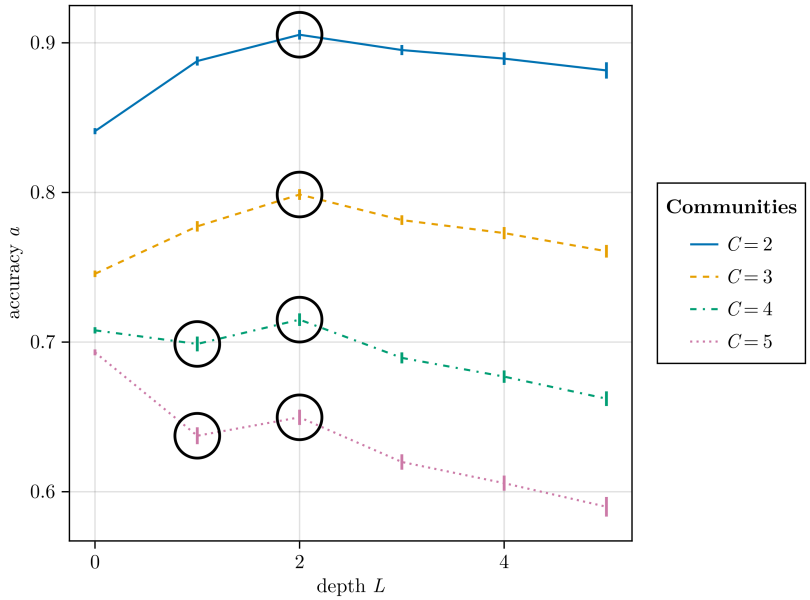

Figure 4: Performance curves can have multiple local extrema (circled in black)
Parameters: $N = 100$, $p_{\text{in}} = 0.05$, $p_{\text{out}} = 0.01$, $\sigma = 0.5$. Estimation: $G = 100$, $T = 20$.

is sometimes necessary to observe such phenomena: with our choice of parameters, the curves for $C < 4$ display a more familiar shape with a single peak.

5 Conclusion

Our main contribution consists in numerically tractable estimates for the Bayes classification accuracy of a linearized GCN on the CSBM. These estimates come from a Gaussian mixture formulation of the embedding distribution, which we can fully compute for $L = 1$ and efficiently approximate for $L > 1$. We hope that they can be useful to researchers seeking better intuition on the smoothing effects of graph convolutions.

The major caveat is that our theorems only hold for the linearized version of the GCN, without activation functions. Nonlinearities would destroy the Gaussian nature of the mixture components, making it much harder to analyze them with the tools we present. Our setting also complicates comparison with the statistical physics literature [14, 17, 18, 31]. We do not let N or D tend to infinity, our community means are fixed instead of being drawn at random, and most importantly we limit ourselves to GCN-based community detection with an arbitrary classification layer instead of studying the true Bayes-optimal algorithm.

After one layer, Theorem 1 gives explicit mixture expressions, but we did not seek a way to leverage these expressions analytically (we relied on numerical quadrature instead). The hurdle we face is the lack of formulas for the classification error even between two single Gaussians, if they are multivariate and not isotropic. As a result, classification error between more than two distributions, each of which is a mixture of Gaussians, is probably out of reach from exact derivations. Still, it may be worth looking into information-theoretical bounds: such a perspective is briefly outlined in Appendix C.

Beyond one layer, the method inspired by Theorem 2 remains dependent on Monte-Carlo integration. It is theoretically possible to obtain samples of the sufficient statistics $(\widetilde{\boldsymbol{s}}_u, \widetilde{\boldsymbol{t}}_u)$ one by one, paying a time and space cost that does not scale with N. Indeed, simulating a length-L random walk on a graph does not require sampling the whole graph, since it is a local process. However, we found it more efficient to sample the whole graph and obtain several values of $(\widetilde{\boldsymbol{s}}_u, \widetilde{\boldsymbol{t}}_u)$ at once, which are unfortunately correlated. It remains an open question to quantify the impact of this correlation on the

error estimates. More broadly, we should conduct a proper statistical study on the precision of these estimators as a function of graph parameters.

We are also curious about an extension of our approach to continuous-time settings, where the number of layers is no longer constrained to be an integer. Even if the first convolutional layer worsens performance, perhaps the optimal depth could be a non-integer value. This corresponds to a paradigm shift from graph random walk to graph diffusion, and we leave it for future study.

Acknowledgements

This research was supported by the Swiss National Science Foundation grant SNFS number 200021-182407. This research was supported by the Swiss National Science Foundation grant SNFS OperaGOST number 200021 200390.

We thank our colleagues for fruitful discussions, especially Odilon Duranthon.

References

[1] Emmanuel Abbe. "Community Detection and Stochastic Block Models: Recent Developments". In: *Journal of Machine Learning Research* 18.177 (2018), pp. 1–86. ISSN: 1533-7928. URL: http://jmlr.org/papers/v18/16-480.html. 2

[2] Aseem Baranwal, Kimon Fountoulakis, and Aukosh Jagannath. "Effects of Graph Convolutions in Multi-layer Networks". en. In: The Eleventh International Conference on Learning Representations. Feb. 1, 2023. URL: https://openreview.net/forum?id=P-73JPgRs0R. 2

[3] Aseem Baranwal, Kimon Fountoulakis, and Aukosh Jagannath. "Graph Convolution for Semi-Supervised Classification: Improved Linear Separability and Out-of-Distribution Generalization". en. In: *Proceedings of the 38th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, July 1, 2021, pp. 684–693. URL: https://proceedings.mlr.press/v139/baranwal21a.html. 2

[4] Aseem Baranwal, Aukosh Jagannath, and Kimon Fountoulakis. *Optimality of Message-Passing Architectures for Sparse Graphs*. May 17, 2023. DOI: 10.48550/arXiv.2305.10391. arXiv: 2305.10391 [cs, stat]. URL: http://arxiv.org/abs/2305.10391. Pre-published. 2

[5] Jeff Bezanson et al. "Julia: A Fresh Approach to Numerical Computing". en. In: *SIAM Review* 59.1 (Jan. 2017), pp. 65–98. ISSN: 0036-1445, 1095-7200. DOI: 10.1137/141000671. URL: https://epubs.siam.org/doi/10.1137/141000671. 6

[6] Joan Bruna and Xiang Li. "Community Detection with Graph Neural Networks". In: (May 23, 2017). URL: https://www.researchgate.net/profile/Joan-Bruna/publication/317088067_Community_Detection_with_Graph_Neural_Networks/links/595a7396a6fdcc36b4d7b544/Community-Detection-with-Graph-Neural-Networks.pdf. 2

[7] Chen Cai and Yusu Wang. *A Note on Over-Smoothing for Graph Neural Networks*. June 23, 2020. DOI: 10.48550/arXiv.2006.13318. arXiv: 2006.13318 [cs, stat]. URL: http://arxiv.org/abs/2006.13318. Pre-published. 2

[8] Ines Chami et al. "Machine Learning on Graphs: A Model and Comprehensive Taxonomy". In: *Journal of Machine Learning Research* 23.89 (2022), pp. 1–64. ISSN: 1533-7928. URL: http://jmlr.org/papers/v23/20-852.html. 1

[9] Ming Chen et al. "Simple and Deep Graph Convolutional Networks". en. In: *Proceedings of the 37th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, Nov. 21, 2020, pp. 1725–1735. URL: https://proceedings.mlr.press/v119/chen20v.html. 2

[10] T. M. Cover and Joy A. Thomas. *Elements of Information Theory*. 2nd ed. Hoboken, N.J: Wiley-Interscience, 2006. 748 pp. ISBN: 978-0-471-24195-9. Google Books: VWq5GG6ycxMC. 17

[11] Caleb Dahlke and Jason Pacheco. "On Convergence of Polynomial Approximations to the Gaussian Mixture Entropy". en. In: *Advances in Neural Information Processing Systems* 36 (Dec. 15, 2023), pp. 75469–75490. URL: https://proceedings.neurips.cc/paper_files/paper/2023/hash/ee860a9fa65a55a335754c557a5211de-Abstract-Conference.html. 17

[12] Guillaume Dalle. *Gdalle/LoG24-GCN-ContextualSBM: V2 - LoG Camera-Ready*. Version v2-log-final. Zenodo, Nov. 22, 2024. DOI: 10.5281/zenodo.14204660. URL: https://zenodo.org/records/14204660. 6

[13] Simon Danisch and Julius Krumbiegel. "Makie.Jl: Flexible High-Performance Data Visualization for Julia". en. In: *Journal of Open Source Software* 6.65 (Sept. 1, 2021), p. 3349. ISSN: 2475-9066. DOI: 10.21105/joss.03349. URL: https://joss.theoj.org/papers/10.21105/joss.03349. 15

[14] Yash Deshpande et al. "Contextual Stochastic Block Models". In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/08fc80de8121419136e443a70489c123-Paper.pdf. 2, 9

[15] Francesco Di Giovanni et al. *Graph Neural Networks as Gradient Flows: Understanding Graph Convolutions via Energy*. Oct. 8, 2022. DOI: 10.48550/arXiv.2206.10991. arXiv: 2206.10991 [cs, stat]. URL: http://arxiv.org/abs/2206.10991. Pre-published. 1

[16] Yijun Ding and Amit Ashok. "Bounds on Mutual Information of Mixture Data for Classification Tasks". EN. In: *JOSA A* 39.7 (July 1, 2022), pp. 1160–1171. ISSN: 1520-8532. DOI: 10.1364/JOSAA.456861. URL: https://opg.optica.org/josaa/abstract.cfm?uri=josaa-39-7-1160. 17

[17] O. Duranthon and L. Zdeborová. *Asymptotic Generalization Error of a Single-Layer Graph Convolutional Network*. Mar. 20, 2024. DOI: 10.48550/arXiv.2402.03818. arXiv: 2402.03818. URL: http://arxiv.org/abs/2402.03818. Pre-published. 2, 9

[18] O. Duranthon and L. Zdeborová. *Optimal Inference in Contextual Stochastic Block Models*. June 6, 2023. DOI: 10.48550/arXiv.2306.07948. arXiv: 2306.07948 [cs]. URL: http://arxiv.org/abs/2306.07948. Pre-published. 2, 9

[19] J.-L. Durrieu, J.-Ph. Thiran, and F. Kelly. "Lower and Upper Bounds for Approximation of the Kullback-Leibler Divergence between Gaussian Mixture Models". In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Mar. 2012, pp. 4833–4836. DOI: 10.1109/ICASSP.2012.6289001. URL: https://ieeexplore.ieee.org/abstract/document/6289001?casa_token=muL8U9RzLYcAAAAA:tXeLQPRgxFUWboqBco-t3Kt_W-Ywsu8g8-NmpLRLcc8v_MNFCN8YHA-QFcEPJHUafzKap85iLA. 17

[20] Federico Errica. "On Class Distributions Induced by Nearest Neighbor Graphs for Node Classification of Tabular Data". en. In: *Advances in Neural Information Processing Systems* 36 (Dec. 15, 2023), pp. 28910–28940. URL: https://proceedings.neurips.cc/paper_files/paper/2023/hash/5c1863f711c721648387ac2ef745facb-Abstract-Conference.html. 3, 4

[21] Justin Gilmer et al. "Neural Message Passing for Quantum Chemistry". en. In: *Proceedings of the 34th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, July 17, 2017, pp. 1263–1272. URL: https://proceedings.mlr.press/v70/gilmer17a.html. 1

[22] M. Hellman and J. Raviv. "Probability of Error, Equivocation, and the Chernoff Bound". In: *IEEE Transactions on Information Theory* 16.4 (July 1970), pp. 368–372. ISSN: 1557-9654. DOI: 10.1109/TIT.1970.1054466. URL: https://ieeexplore.ieee.org/abstract/document/1054466. 17

[23] Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. "Stochastic Blockmodels: First Steps". In: *Social Networks* 5.2 (June 1, 1983), pp. 109–137. ISSN: 0378-8733. DOI: 10.1016/0378-8733(83)90021-7. URL: https://www.sciencedirect.com/science/article/pii/0378873383900217. 2

[24] Steven G. Johnson. *QuadGK.Jl: Gauss–Kronrod Integration in Julia*. 2013. URL: https://github.com/JuliaMath/QuadGK.jl. 15

[25] Steven G. Johnson. *The HCubature.Jl Package for Multi-Dimensional Adaptive Integration in Julia*. 2017. URL: https://github.com/JuliaMath/HCubature.jl. 15

[26] *JuliaAI/MLJLinearModels.Jl*. JuliaAI, Nov. 9, 2023. URL: https://github.com/JuliaAI/MLJLinearModels.jl. 15

[27] Nicolas Keriven. "Not Too Little, Not Too Much: A Theoretical Analysis of Graph (over)Smoothing". In: *Advances in Neural Information Processing Systems*. Vol. 35. Curran Associates, Inc., 2022, pp. 2268–2281. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/0f956ca6f667c62e0f71511773c86a59-Paper-Conference.pdf. 1–3, 7

[28] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". en. In: The Fifth International Conference on Learning Representations. 2017. URL: https://openreview.net/forum?id=SJU4ayYgl. 1

[29] Artemy Kolchinsky and Brendan D. Tracey. "Estimating Mixture Entropy with Pairwise Distances". en. In: *Entropy* 19.7 (7 July 2017), p. 361. ISSN: 1099-4300. DOI: 10.3390/e19070361. URL: https://www.mdpi.com/1099-4300/19/7/361. 17

[30] Qimai Li, Zhichao Han, and Xiao-Ming Wu. "Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning". en. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1 (1 Apr. 29, 2018). ISSN: 2374-3468. DOI: 10.1609/aaai.v32i1.11604. URL: https://ojs.aaai.org/index.php/AAAI/article/view/11604. 2

[31] Chen Lu and Subhabrata Sen. "Contextual Stochastic Block Model: Sharp Thresholds and Contiguity". In: *Journal of Machine Learning Research* 24.54 (2023), pp. 1–34. ISSN: 1533-7928. URL: http://jmlr.org/papers/v24/20-1419.html. 2, 9

[32] Yao Ma et al. "Is Homophily a Necessity for Graph Neural Networks?" en. In: International Conference on Learning Representations. Jan. 28, 2022. URL: https://openreview.net/forum?id=ucASPPD9GKN. 7

[33] Patrick K. Mogensen and Asbjørn N. Riseth. "Optim: A Mathematical Optimization Package for Julia". en. In: *Journal of Open Source Software* 3.24 (Apr. 5, 2018), p. 615. ISSN: 2475-9066. DOI: 10.21105/joss.00615. URL: https://joss.theoj.org/papers/10.21105/joss.00615. 16

[34] Hoang NT and Takanori Maehara. *Revisiting Graph Neural Networks: All We Have Is Low-Pass Filters*. May 26, 2019. DOI: 10.48550/arXiv.1905.09550. arXiv: 1905.09550 [cs, math, stat]. URL: http://arxiv.org/abs/1905.09550. Pre-published. 2

[35] Kenta Oono and Taiji Suzuki. "Graph Neural Networks Exponentially Lose Expressive Power for Node Classification". en. In: International Conference on Learning Representations. Oct. 20, 2020. URL: https://openreview.net/forum?id=S1ldO2EFPr. 2

[36] Andreas Roth. *Simplifying the Theory on Over-Smoothing*. Sept. 2, 2024. DOI: 10.48550/arXiv.2407.11876. arXiv: 2407.11876 [cs]. URL: http://arxiv.org/abs/2407.11876. Pre-published. 2

[37] T. Konstantin Rusch, Michael M. Bronstein, and Siddhartha Mishra. *A Survey on Oversmoothing in Graph Neural Networks*. Mar. 20, 2023. DOI: 10.48550/arXiv.2303.10993. arXiv: 2303.10993 [cs]. URL: http://arxiv.org/abs/2303.10993. Pre-published. 1

[38] F. Scarselli et al. "The Graph Neural Network Model". In: *IEEE Transactions on Neural Networks* 20.1 (Jan. 2009), pp. 61–80. ISSN: 1045-9227, 1941-0093. DOI: 10.1109/TNN.2008.2005605. URL: http://ieeexplore.ieee.org/document/4700287/. 1

[39] Cheng Shi et al. *Statistical Mechanics of Generalization In Graph Convolution Networks*. Dec. 26, 2022. DOI: 10.48550/arXiv.2212.13069. arXiv: 2212.13069 [cond-mat, stat]. URL: http://arxiv.org/abs/2212.13069. Pre-published. 2

[40] Terence Tao. *An Improvement to Bennett's Inequality for the Poisson Distribution*. en. What's new. Dec. 13, 2022. URL: https://terrytao.wordpress.com/2022/12/13/an-improvement-to-bennetts-inequality-for-the-poisson-distribution/. 6

[41] Petar Veličković. *Message Passing All the Way Up*. Feb. 22, 2022. arXiv: 2202.11097 [cs, stat]. URL: http://arxiv.org/abs/2202.11097. Pre-published. 2

[42] Robert Wang, Aseem Baranwal, and Kimon Fountoulakis. *Analysis of Corrected Graph Convolutions*. May 22, 2024. DOI: 10.48550/arXiv.2405.13987. arXiv: 2405.13987 [cs, math, stat]. URL: http://arxiv.org/abs/2405.13987. Pre-published. 1, 3

[43] Felix Wu et al. "Simplifying Graph Convolutional Networks". en. In: *Proceedings of the 36th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, May 24, 2019, pp. 6861–6871. URL: https://proceedings.mlr.press/v97/wu19e.html. 1

[44] Xinyi Wu et al. "A Non-Asymptotic Analysis of Oversmoothing in Graph Neural Networks". en. In: The Eleventh International Conference on Learning Representations. 2023. URL: https://openreview.net/forum?id=CJd-BtnwtXq. 1–3

A Proofs

A.1 Bayes accuracy

Given an embedding $\boldsymbol{h}$, maximum likelihood classification involves finding the community $\widehat{y}$ whose density evaluated at $\boldsymbol{h}$ is highest. We compute the Bayes accuracy as the probability of $\widehat{y}$ being equal to the true community y:

$$a = \mathbb{P}(\widehat{y} = y) = \sum_{c=1}^{C} \mathbb{P}(y = c)\mathbb{P}(\widehat{y} = c \mid y = c) \tag{15}$$

$$= \sum_{c=1}^{C} f_c \int \pi_c(\boldsymbol{h})\, \mathbf{1}\{f_c \pi_c(\boldsymbol{h}) = \max_{c_1} f_{c_1} \pi_{c_1}(\boldsymbol{h})\}\, \mathrm{d}\boldsymbol{h} \tag{16}$$

$$= \int_{\mathbb{R}^D} \max_{c_1} f_{c_1} \pi_{c_1} \underbrace{\sum_{c=1}^{C} \mathbf{1}\{f_c \pi_c = \max_{c_1} f_{c_1} \pi_{c_1}\}}_{1}. \tag{17}$$

Even though it is not obvious from its integral expression in Equation (4), the definition of a as a probability ensures that it belongs to $[0, 1]$.

A.2 Theorem 1

After a single layer, the embedding of $u \in c$ is deduced from Equation (1):

$$\boldsymbol{h}_u = \frac{1}{\widetilde{d}_u} \sum_{v=1}^{N} \widetilde{A}_{u,v} \boldsymbol{x}_v = \frac{1}{\widetilde{d}_u} \sum_{c_1=1}^{C} \sum_{v \in c_1} \widetilde{A}_{u,v} \boldsymbol{x}_v. \tag{18}$$

Conditioned on the graph, $\boldsymbol{h}_u$ is a fixed linear transformation of $\boldsymbol{X}$, which implies that $\mathbb{P}(\boldsymbol{h}_u \mid \boldsymbol{A})$ is a Gaussian. Its first two conditional moments are given by

$$\mathbb{E}[\boldsymbol{h}_u \mid \boldsymbol{A}] = \frac{1}{\widetilde{d}_u} \sum_{c_1=1}^{C} \sum_{v \in c_1} \widetilde{A}_{u,v} \mathbb{E}[\boldsymbol{x}_v] \tag{19}$$

$$\mathbb{E}[\boldsymbol{h}_u \boldsymbol{h}_u^\top \mid \boldsymbol{A}] = \frac{1}{\widetilde{d}_u^2} \sum_{c_1=1}^{C} \sum_{c_2=1}^{C} \sum_{v \in c_1} \sum_{w \in c_2} \widetilde{A}_{u,v} \widetilde{A}_{u,w}\, \mathbb{E}[\boldsymbol{x}_v \boldsymbol{x}_w^\top] \tag{20}$$

The key idea is to exploit the symmetries of the CSBM. Indeed, the expectations $\mathbb{E}[\boldsymbol{x}_v]$ and $\mathbb{E}[\boldsymbol{x}_v \boldsymbol{x}_w^\top]$ do not depend on the specific choice of vertices (v, w), only on their communities (c_1, c_2). The one exception is when $v = w$:

$$\mathbb{E}[\boldsymbol{x}_v] = \boldsymbol{\mu}_{c_1} \qquad \text{and} \qquad \mathbb{E}[\boldsymbol{x}_v \boldsymbol{x}_w^\top] = \begin{cases} \boldsymbol{\mu}_{c_1} (\boldsymbol{\mu}_{c_2})^\top & \text{if } v \neq w \\ \boldsymbol{\mu}_{c_1} (\boldsymbol{\mu}_{c_1})^\top + \boldsymbol{\Sigma}_{c_1} & \text{if } v = w. \end{cases} \tag{21}$$

For the conditional first moment, we combine Equations (19) and (21):

$$\mathbb{E}[\boldsymbol{h}_u \mid \boldsymbol{A}] = \frac{1}{\widetilde{d}_u} \sum_{c_1} \sum_{v \in c_1} \widetilde{A}_{u,v} \mathbb{E}[\boldsymbol{x}_v] \tag{22}$$

$$= \frac{1}{\widetilde{d}_u} \sum_{c_1} \left(\sum_{v \in c_1} \widetilde{A}_{u,v} \right) \boldsymbol{\mu}_{c_1} \tag{23}$$

$$= \frac{1}{\widetilde{d}_u} \sum_{c_1} \widetilde{d}_{u,c_1} \boldsymbol{\mu}_{c_1}. \tag{24}$$

For the conditional second moment, we split Equation (20) in two depending on whether $v = w$, then invoke Equation (21):

$$\mathbb{E}[\boldsymbol{h}_u \boldsymbol{h}_u^\top \mid \boldsymbol{A}] = \frac{1}{\widetilde{d}_u^2} \sum_{c_1,c_2} \sum_{\substack{v \in c_1 \\ w \in c_2}} \widetilde{A}_{u,v} \widetilde{A}_{u,w} \, \mathbb{E}[\boldsymbol{x}_v \boldsymbol{x}_w^\top] \tag{25}$$

$$= \frac{1}{\widetilde{d}_u^2} \sum_{c_1,c_2} \sum_{\substack{v \in c_1 \\ w \in c_2 \\ v \neq w}} \widetilde{A}_{u,v} \widetilde{A}_{u,w} \, \mathbb{E}[\boldsymbol{x}_v \boldsymbol{x}_w^\top] + \frac{1}{\widetilde{d}_u^2} \sum_{c_1,c_2} \sum_{\substack{v \in c_1 \\ w \in c_2 \\ v = w}} \widetilde{A}_{u,v} \widetilde{A}_{u,w} \, \mathbb{E}[\boldsymbol{x}_v \boldsymbol{x}_w^\top] \tag{26}$$

$$= \frac{1}{\widetilde{d}_u^2} \sum_{c_1,c_2} \sum_{\substack{v \in c_1 \\ w \in c_2 \\ v \neq w}} \widetilde{A}_{u,v} \widetilde{A}_{u,w} \, \boldsymbol{\mu}_{c_1} (\boldsymbol{\mu}_{c_2})^\top + \frac{1}{\widetilde{d}_u^2} \sum_{c_1,c_2} \sum_{\substack{v \in c_1 \\ w \in c_2 \\ v = w}} \widetilde{A}_{u,v} \widetilde{A}_{u,w} \left(\boldsymbol{\mu}_{c_1} (\boldsymbol{\mu}_{c_1})^\top + \boldsymbol{\Sigma}_{c_1} \right). \tag{27}$$

We rearrange the terms to observe

$$\mathbb{E}[\boldsymbol{h}_u \boldsymbol{h}_u^\top \mid \boldsymbol{A}] = \frac{1}{\widetilde{d}_u^2} \sum_{c_1,c_2} \sum_{\substack{v \in c_1 \\ w \in c_2}} \widetilde{A}_{u,v} \widetilde{A}_{u,w} \, \boldsymbol{\mu}_{c_1} (\boldsymbol{\mu}_{c_2})^\top + \frac{1}{\widetilde{d}_u^2} \sum_{\substack{c_1,c_2 \\ c_1 = c_2}} \sum_{v \in c_1} \widetilde{A}_{u,v}^2 \, \boldsymbol{\Sigma}_{c_1} \tag{28}$$

$$= \left(\frac{1}{\widetilde{d}_u} \sum_{c_1} \sum_{v \in c_1} \widetilde{A}_{u,v} \, \boldsymbol{\mu}_{c_1} \right) \left(\frac{1}{\widetilde{d}_u} \sum_{c_2} \sum_{w \in c_2} \widetilde{A}_{u,w} \, \boldsymbol{\mu}_{c_2} \right)^\top + \frac{1}{\widetilde{d}_u^2} \sum_{c_1} \left(\sum_{v \in c_1} \widetilde{A}_{u,v}^2 \right) \boldsymbol{\Sigma}_{c_1}. \tag{29}$$

On the left, we recognize the square of the first moment $\mathbb{E}[\boldsymbol{h}_u \mid \boldsymbol{A}]$ computed earlier. On the right, we exploit the fact that adjacency matrices are binary (and so are augmented adjacency matrices for lack of self-loops), which implies $\widetilde{A}_{u,v}^2 = \widetilde{A}_{u,v}$. Therefore,

$$\mathbb{E}[\boldsymbol{h}_u \boldsymbol{h}_u^\top \mid \boldsymbol{A}] = \mathbb{E}[\boldsymbol{h}_u \mid \boldsymbol{A}] \, \mathbb{E}[\boldsymbol{h}_u \mid \boldsymbol{A}]^\top + \frac{1}{\widetilde{d}_u^2} \sum_{c_1} \left(\sum_{v \in c_1} \widetilde{A}_{u,v} \right) \boldsymbol{\Sigma}_{c_1} \tag{30}$$

$$= \mathbb{E}[\boldsymbol{h}_u \mid \boldsymbol{A}] \, \mathbb{E}[\boldsymbol{h}_u \mid \boldsymbol{A}]^\top + \frac{1}{\widetilde{d}_u^2} \sum_{c_1} \widetilde{d}_{u,c_1} \, \boldsymbol{\Sigma}_{c_1}. \tag{31}$$

We have thus computed the conditional variance:

$$\mathbb{V}[\boldsymbol{h}_u \mid \boldsymbol{A}] = \mathbb{E}[\boldsymbol{h}_u \boldsymbol{h}_u^\top \mid \boldsymbol{A}] - \mathbb{E}[\boldsymbol{h}_u \mid \boldsymbol{A}] \, \mathbb{E}[\boldsymbol{h}_u \mid \boldsymbol{A}]^\top = \frac{1}{\widetilde{d}_u^2} \sum_{c_1} \widetilde{d}_{u,c_1} \boldsymbol{\Sigma}_{c_1}. \tag{32}$$

To sum up Equations (24) and (32), the conditional distribution $\mathbb{P}(\boldsymbol{h}_u \mid \boldsymbol{A})$ is a Gaussian whose parameters are positive combinations of the initial community parameters:

$$\mathbb{E}[\boldsymbol{h}_u \mid \boldsymbol{A}] = \frac{1}{\widetilde{d}_u} \sum_{c_1} \widetilde{d}_{u,c_1} \boldsymbol{\mu}_{c_1} \qquad \text{and} \qquad \mathbb{V}[\boldsymbol{h}_u \mid \boldsymbol{A}] = \frac{1}{\widetilde{d}_u^2} \sum_{c_1} \widetilde{d}_{u,c_1} \boldsymbol{\Sigma}_{c_1}. \tag{33}$$

Crucially, $\mathbb{E}[\boldsymbol{h}_u \mid \boldsymbol{A}]$ and $\mathbb{V}[\boldsymbol{h}_u \mid \boldsymbol{A}]$ do not depend on the whole graph $\boldsymbol{A}$, only on the vector of community degrees $\boldsymbol{d}_u$ (from which we can deduce the augmented version $\widetilde{\boldsymbol{d}}_u$):

$$\mathbb{E}[\boldsymbol{h}_u \mid \boldsymbol{A}] = \mathbb{E}[\boldsymbol{h}_u \mid \boldsymbol{d}_u] \qquad \text{and} \qquad \mathbb{V}[\boldsymbol{h}_u \mid \boldsymbol{A}] = \mathbb{V}[\boldsymbol{h}_u \mid \boldsymbol{d}_u] \tag{34}$$

By independence between edges in an SBM, each community degree d_{u,c_1} follows a Binomial distribution. Recall that vertex u belongs to community c, so it has N_{c,c_1} possible neighbors in community c_1. Each one of these possible neighbors in c_1 is connected to u with the same probability q_{c,c_1}. Accordingly, the community degree d_{u,c_1} follows a Binomial distribution $\mathcal{B}(N_{c,c_1}, q_{c,c_1})$, independently from other community degrees d_{u,c_2}. We can write the joint distribution of the vector of community degrees as a product distribution:

$$\mathbb{P}(\boldsymbol{d}_u) = \prod_{c_1=1}^{C} \mathcal{B}(N_{c,c_1}, q_{c,c_1})[d_{u,c_1}] \tag{35}$$

In conclusion, the unconditional distribution $\mathbb{P}(\boldsymbol{h}_u)$ is a mixture of Gaussians with one component per value of the community degree vector $\boldsymbol{d}_u$, and we can exactly describe the mixture components and their weights:

$$\mathbb{P}(\boldsymbol{h}_u) = \sum_{\boldsymbol{d}_u} \mathbb{P}(\boldsymbol{d}_u)\, \mathcal{N}\big(\mathbb{E}[\boldsymbol{h}_u \mid \boldsymbol{d}_u], \mathbb{V}[\boldsymbol{h}_u \mid \boldsymbol{d}_u]\big) \tag{36}$$

This completes the proof of Theorem 1.

A.3 Theorem 2

We define the matrix $\widetilde{\boldsymbol{R}} := (\widetilde{\boldsymbol{D}}^{-1}\widetilde{\boldsymbol{A}})^L$ so that $\boldsymbol{H} = \widetilde{\boldsymbol{R}}\boldsymbol{X}$. For $L = 1$ we have $\widetilde{\boldsymbol{R}} = \widetilde{\boldsymbol{D}}^{-1}\widetilde{\boldsymbol{A}}$, which means $\widetilde{R}_{u,v} = \widetilde{A}_{u,v}/\widetilde{d}_u$. For $L > 1$, there is no such simple formula, but we can still use Equation (1) to write an analogue of Equation (18):

$$\boldsymbol{h}_u = \sum_{v=1}^{N} \widetilde{R}_{u,v}\boldsymbol{x}_v = \sum_{c_1=1}^{C} \sum_{v \in c_1} \widetilde{R}_{u,v}\boldsymbol{x}_v. \tag{37}$$

From there, we retrace the same steps as in Section A.2 and obtain the following expressions, which mirror Equations (23) and (29):

$$\mathbb{E}[\boldsymbol{h}_u \mid \boldsymbol{A}] = \sum_{c_1} \left(\sum_{v \in c_1} \widetilde{R}_{u,v} \right) \boldsymbol{\mu}_{c_1} \tag{38}$$

$$\mathbb{V}[\boldsymbol{h}_u \mid \boldsymbol{A}] = \sum_{c_1} \left(\sum_{v \in c_1} \widetilde{R}^2_{u,v} \right) \boldsymbol{\mu}_{c_1} \tag{39}$$

This prompts us to define generalizations of the community degrees:

$$\widetilde{s}_{u,c_1} := \sum_{v \in c_1} \widetilde{R}_{u,v} \qquad \text{and} \qquad \widetilde{t}_{u,c_1} := \sum_{v \in c_1} \widetilde{R}^2_{u,v}. \tag{40}$$

Beyond one layer, we capture all the relevant information about $\boldsymbol{h}_u$ in the graph $\boldsymbol{A}$ by conditioning not on a single vector $\boldsymbol{d}_u$, but on a pair of vectors $(\widetilde{\boldsymbol{s}}_u, \widetilde{\boldsymbol{t}}_u)$. Equation (34) then becomes

$$\mathbb{E}[\boldsymbol{h}_u \mid \boldsymbol{A}] = \mathbb{E}[\boldsymbol{h}_u \mid \widetilde{\boldsymbol{s}}_u, \widetilde{\boldsymbol{t}}_u] \qquad \text{and} \qquad \mathbb{V}[\boldsymbol{h}_u \mid \boldsymbol{A}] = \mathbb{V}[\boldsymbol{h}_u \mid \widetilde{\boldsymbol{s}}_u, \widetilde{\boldsymbol{t}}_u]. \tag{41}$$

This completes the proof of Theorem 2.

B Experimental details

Our code uses Julia version 1.11.1 with the following main packages: `SparseArrays.jl` for sparse linear algebra, `QuadGK.jl` [24] and `HCubature.jl` [25] for numerical integration, `MLJLinearModels.jl` [26] for logistic regression, `Makie.jl` [13] for visualization, `Pluto.jl` for experiment notebooks.

B.1 Protocol

For single-layer analyses, we estimate the Bayes accuracy by numerically integrating Equation (4) using the truncated mixtures $\widehat{\pi}_c$ from Equation (13), with a relative tolerance $\texttt{rtol} = 10^{-5}$ for the quadrature.

For multi-layer analyses, we switch to Monte-Carlo integration of Equation (4) using the Monte-Carlo mixture estimates $\widehat{\pi}_c$ from Equation (14) (there are two layers of Monte-Carlo sampling). We draw G independent graphs of the same size N. Each of those graphs gives us N_c samples of $(\widetilde{\boldsymbol{s}}_u, \widetilde{\boldsymbol{t}}_u)$ for each community c, which we use to approximate $\widehat{\pi}_c$. Then, we evaluate the Bayes accuracy using a total of GN samples from $\sum_c f_c \widehat{\pi}_c$. This whole process is repeated T times to provide uncertainty estimates (error bars are $\pm$ one standard deviation over these T values).

B.2 Additional illustrations

On Figure 5 we display a two-dimensional visualization of the mixture distributions (Figure 5), which did not fit in the main paper.

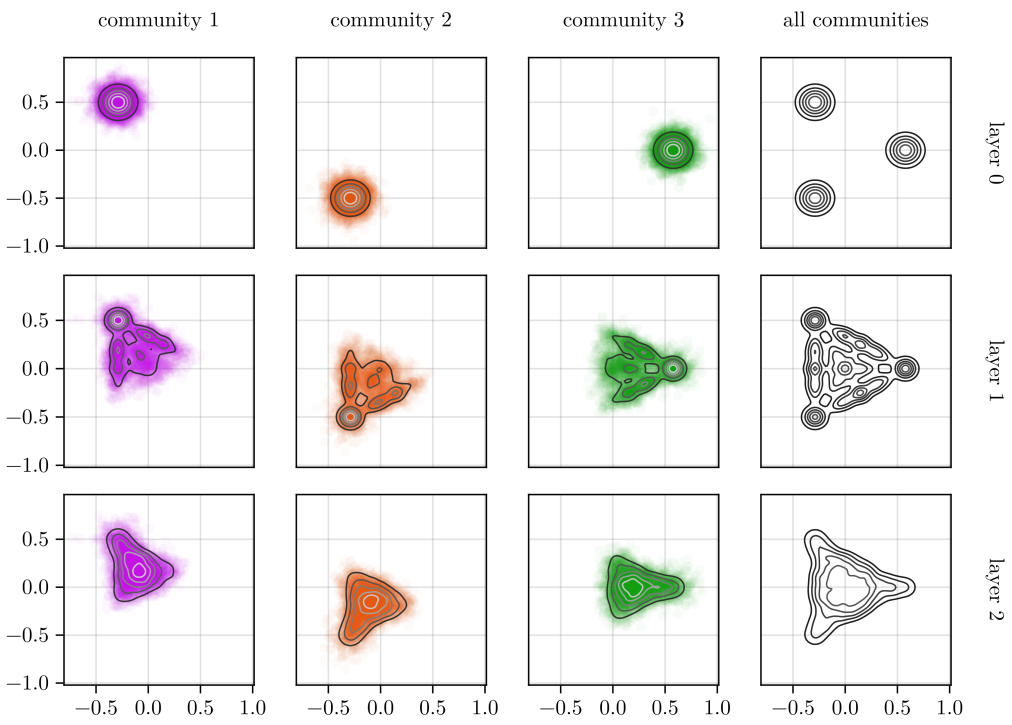

Figure 5: Embedding distributions in 2D are Gaussian mixtures.
Dots are empirical samples, contours represent our density estimates.
Parameters: $N = 100$, $C = 3$, $p_{\text{in}} = 0.03$, $p_{\text{out}} = 0.02$, $\sigma = 0.1$. Estimation: $G = 100$.

B.3 Validation against logistic regression

We consider the same multi-layer experimental setting as in Section 4.4. Due to the structure of the Circular 2D test case and the overall symmetry of the SBM associated with it, we expect the decision boundaries to look like C clock quadrants meeting at the origin (at least in the first few layers, so long as the variance is low enough). Therefore, it seems reasonable that multinomial logistic regression would attain near-optimal performance. To verify that our mixture-based estimators are correct, we therefore compare them with the predictions of a linearized GCN whose final classifier is a multinomial logistic regression. This final classifier is trained and evaluated on the same number of samples that is used to compute and evaluate the mixtures.

The hyperparameters and training algorithm correspond to the default settings of `MultinomialRegression` from `MLJLinearModels.jl`. More precisely, the loss is the standard multinomial logistic loss, where the L1 and L2 regularizations λ and γ are both set to 0 and no intercept is fitted. The optimization algorithm is `LBFGS` as implemented in `Optim.jl` [33], with a modified relative tolerance of 10^{-4} in the objective value. The other hyperparameters of `LBFGS` keep their default values from `Optim.jl`: the most important ones are the limited memory size $m = 10$, the step size chosen by Hager-Zhang line search, the maximum of 1000 approximate Newton iterations and the absence of time limit. Given that the logistic loss is convex, the choice of hyperparameters has little effect on the convergence of the optimization algorithm, which is why we do not tune them.

As we can see in Table 1 there is good alignment between the empirical accuracy of the linearized GCN with logistic regression and the one we deduce from Monte-Carlo mixture estimates.

Table 1: Mixture estimators for accuracy are coherent with logistic regression
Parameters: $N = 100$, $p_{\text{in}} = 0.05$, $p_{\text{out}} = 0.01$, $\sigma = 0.5$. Estimation: $G = 100$, $T = 20$.

comm.	method	estimated accuracy			
		depth $L = 0$	depth $L = 1$	depth $L = 2$	depth $L = 3$
$C = 2$	mixtures	0.841 ± 0.002	0.888 ± 0.003	0.905 ± 0.0033	0.895 ± 0.0034
$C = 2$	logistic	0.841 ± 0.0042	0.89 ± 0.0035	0.908 ± 0.0041	0.898 ± 0.0045
$C = 3$	mixtures	0.746 ± 0.0022	0.777 ± 0.0035	0.799 ± 0.0036	0.782 ± 0.0032
$C = 3$	logistic	0.744 ± 0.0053	0.777 ± 0.0064	0.798 ± 0.0052	0.782 ± 0.0062
$C = 4$	mixtures	0.708 ± 0.0021	0.699 ± 0.0049	0.715 ± 0.0042	0.689 ± 0.0037
$C = 4$	logistic	0.709 ± 0.0058	0.697 ± 0.0063	0.714 ± 0.0061	0.688 ± 0.0082
$C = 5$	mixtures	0.693 ± 0.002	0.637 ± 0.0057	0.65 ± 0.0051	0.62 ± 0.0052
$C = 5$	logistic	0.692 ± 0.0082	0.634 ± 0.01	0.647 ± 0.011	0.616 ± 0.011

C Information-theoretical bounds

Here we sketch a possible way to derive analytical error estimates from the mixtures in Theorem 1. Recall from Section 2 that our goal is to study the best possible estimator (Bayes-optimal) $\widehat{y}$ for the community assignment y of a vertex drawn at random. This estimator must be a measurable function $\varphi^\star$ of the vertex embedding $\boldsymbol{h}$. Its accuracy a is one minus the probability of error, a probability which we denote by $b := 1 - a$.

We use the same notations as in the book of Cover and Thomas [10]. By Fano's inequality [10, Theorem 2.10.1] applied to the Markov chain $y \to \boldsymbol{h} \to \widehat{y}$, the probability of error is linked to the conditional entropy $\mathbb{H}[y|\boldsymbol{h}]$ of the true community given the embedding:

$$\underbrace{\mathbb{H}[\mathcal{B}(b)] + b \log C}_{\psi_C(b)} \geq \mathbb{H}[y|\boldsymbol{h}]. \tag{42}$$

In Equation (42), we know the number of communities C and the entropy $\mathbb{H}[\mathcal{B}(b)]$ of a Bernoulli distribution:

$$\mathbb{H}[\mathcal{B}(b)] = -b \log b - (1 - b) \log(1 - b). \tag{43}$$

Conversely, Hellman and Raviv [22, Equation 41] prove the upper bound

$$b \leq \frac{1}{2} \mathbb{H}[y|\boldsymbol{h}]. \tag{44}$$

Defining $\psi_C^{-1}(\beta)$ to be the smallest root b of the equation $\psi_C(b) = \beta$ in the interval $[0, 1]$ (there can be either one or two roots), we deduce the following:

$$\psi_C^{-1}(\mathbb{H}[y|\boldsymbol{h}]) \leq b \leq \frac{1}{2} \mathbb{H}[y|\boldsymbol{h}]. \tag{45}$$

We still need to control the conditional entropy $\mathbb{H}[y|\boldsymbol{h}]$. By the chain rule for conditional entropy [10, Theorem 2.2.1], we have

$$\mathbb{H}[y|\boldsymbol{h}] = \mathbb{H}[y, \boldsymbol{h}] - \mathbb{H}[\boldsymbol{h}] \tag{46}$$

$$= \mathbb{H}[y] + \mathbb{H}[\boldsymbol{h}|y] - \mathbb{H}[\boldsymbol{h}]. \tag{47}$$

Each one of these terms has a straightforward interpretation:

$$\mathbb{H}[y] = \mathbb{H}\left[\sum_c f_c \delta_c\right] \qquad \text{entropy of a categorical distribution} \tag{48}$$

$$\mathbb{H}[\boldsymbol{h}] = \mathbb{H}\left[\sum_c f_c \pi_c\right] \qquad \text{entropy of a large Gaussian mixture} \tag{49}$$

$$\mathbb{H}[\boldsymbol{h}|y] = \sum_c f_c \, \mathbb{H}[\pi_c] \qquad \text{average of entropies of small Gaussian mixtures} \tag{50}$$

There is no explicit formula for the entropy of a Gaussian mixture (Equations (49) and (50)), but several approximations exist [11, 16, 19, 29]. Combined with Equation (6), one of these approximations might prove sufficient to obtain precise rates of convergence for the Bayes error b. We leave this investigation for future work.