Inferring community characteristics in labelled networks

Lawrence Tray
Department of Engineering
University of Cambridge
lpt30@cam.ac.uk

Ioannis Kontoyiannis
Statistical Laboratory, DPMMS
University of Cambridge
ik355@cam.ac.uk

Abstract

Labelled networks form a very common and important class of data, naturally appearing in numerous applications in science and engineering. A typical inference goal is to determine how the vertex labels (called features) affect the network's graph structure. A standard approach has been to partition the network into blocks grouped by distinct values of the feature of interest. A block-based random graph model – typically a variant of the stochastic block model (SBM) – is then used to test for evidence of asymmetric behaviour within these feature-based communities. Nevertheless, the resulting communities often do not produce a natural partition of the graph. In this work we introduce a new generative model, the feature-first block model (FFBM), which is more effective at describing vertex-labelled undirected graphs and also facilitates the use of richer queries on labelled networks. We develop a Bayesian framework for inference with this model, and we present a method to efficiently sample from the posterior distribution of the FFBM parameters. The FFBM's structure is kept deliberately simple to retain easy interpretability of the parameter values. We apply the proposed methods to a variety of network data to extract the most important features along which the vertices are partitioned. The main advantages of the proposed approach are that the whole feature-space is used automatically, and features can be rank-ordered implicitly according to impact. Any features that do not significantly impact the high-level structure can be discarded to reduce the problem dimension. In cases where the vertex features available do not readily explain the community structure in the resulting network, the approach detects this and is protected against over-fitting. Results on several real-world datasets illustrate the performance of the proposed methods.

1 Introduction

A somewhat surprising property of many real-world networks is that they exhibit strong community structure. In other words, each node will often belong to a cluster of densely connected nodes. There is high interest in recovering the latent communities from the observed graphs. The inferred communities can be exploited for compression algorithms [1] or used for link prediction in incomplete networks [4] to name but a few applications.

We restrict our analysis to labelled networks. These are graphs where we additionally have information about the properties of each vertex. We shall refer to these vertex properties as features. One of the most common questions we can ask of labelled networks is whether a given vertex feature has an impact on the structure of the graph. To answer this question from a Bayesian perspective we must use a random graph model; the most common form is called the stochastic block model (SBM) [10]. This is a latent variable model where each vertex belongs to a single block and the probability two nodes are connected depends only on the block memberships of each. There have been many variants to this model; the most popular are the mixed-membership stochastic block model (MMSBM) [2]

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

and the overlapping stochastic block model (OSBM) [20]. Effectively, these just extend the model to allow each vertex to belong to multiple blocks simultaneously.

However, a major drawback of these graphical models as applied to labelled networks is that they do not automatically include vertex features in the random graph generation process. Approaches based on graph neural networks [9] that utilise vertex features have been developed but these lack the easy interpretability of the simpler models.

To analyse a labelled network using one of the simple SBM variants, a typical inference procedure would be to partition the graph into blocks grouped by distinct values of the feature of interest. The associated model can then be used to test for evidence of heterogeneous connectivity between the feature-grouped blocks. Nevertheless, this approach is limited in that it can only consider one feature at a time; this means any conclusions drawn highly vulnerable to the presence of confounding variables. Furthermore, considering each feature separately makes it difficult to rank order the features by magnitude of impact. Lastly, the feature-grouped blocks are often an unnatural partition of the graph, leading to a poor model fit. We would instead prefer to partition the graph into its most natural blocks and then determine which features best predict the resulting partition.

With these desiderata in mind, we present a novel framework for modelling labelled networks, which we call the feature-first block model (FFBM). This can be thought of as an extension of the SBM to labelled networks. In the FFBM, we use the features first to generate the latent block membership for each vertex. The latent block membership is therefore a stepping stone rather than a starting point in our analysis. We go on to present an efficient algorithm for sampling from the parameters of the feature-to-block generator. We can interpret the sampled FFBM parameters to determine which features have the largest impact on overall graphical structure. Any features found to be irrelevant can be discarded as a form of dimensionality reduction.

2 Preliminaries

This section defines some preliminary concepts required for the subsequent analysis. We first need a model for community-like structure in a graphical network. For this we adopt the stochastic block model (SBM) - widely used across academia. The premise is that each node in the graph belongs to a unique community called a block. The probability that two nodes are connected depends only on the block memberships of each. Graphs drawn from the SBM ensemble exhibit community structure. Specifically, we will use the microcanonical variant of the SBM, proposed by Peixoto [15]. A paraphrased definition is given below for the non-degree-corrected SBM (NDC-SBM).

Definition 2.1 (Microcanonical NDC-SBM) *Let $N \in \mathbb{Z}^+$ denote the number of vertices in an undirected graph. The block memberships are encoded by a vector b of length N where each entry $b_i \in \{1, 2 \dots B\}$. $B \in \mathbb{Z}^+$ is the number of non-empty blocks. Let e be a $B \times B$ matrix of edge counts between blocks (e_{rs} is number of edges from block r onto block s - or twice that number if $r = s$). For undirected graphs e is symmetric. For a non-degree-corrected stochastic block model (NDC-SBM), we say that the graph A is generated as follows:*

$$A \sim \textit{NDC-SBM}_{MC}(b, e) \tag{1}$$

Where edges are placed uniformly at random but respecting the constraint imposed by e and b. The additional parameters N and B are omitted as they are inferred from the shapes of b and e. If we interpret A as an adjacency matrix, then this constraint can be written formally as: $e_{rs} = \sum_{i,j} A_{ij} \mathbb{1}\{b_i = r\} \mathbb{1}\{b_j = s\}$.

Nevertheless, this formulation does not accept high degree variability within blocks as is typical of real-world data. Indeed, the NDC-SBM favours a partition into high-degree and low-degree nodes rather than clusters of inter-connected nodes. We therefore introduce the degree-corrected SBM (DC-SBM) [15] to circumvent these issues.

Definition 2.2 (Microcanonical DC-SBM) *This is much like the NDC-SBM but has an additional parameter k which is an N-length vector encoding the degree sequence (k_i is the degree of vertex i). Therefore, we write:*

$$A \sim \textit{DC-SBM}_{MC}(b, e, k) \tag{2}$$

Once again, edges are placed uniformly at random but respecting the constraints imposed by the parameters. The DC-SBM has the additional constraint that $k_i = \sum_j A_{ij}$. In what follows, we will always assume the degree-corrected model unless otherwise specified.

3 Feature-first block model

In this section we propose a novel generative model for modelling labelled networks. We restrict our analysis to labelled, undirected graphs with N nodes. We define the vector $x_i \in \mathcal{X}^D$ as the feature vector for the i'th vertex. Each vertex has D total features and we assume all entries take values from the same set $\mathcal{X}$. For the majority of datasets we analyse, we deal with binary feature flags so $\mathcal{X} = \{0, 1\}$. The feature vectors $\{x_i\}_{i=1}^N$ are subsumed into the $N \times D$ matrix X.

The proposed generative model (which we call the feature-first block model - FFBM) is given in figure 1. We start, with the feature matrix X and generate a vector of block memberships b. The parameters of this step are encapsulated by θ. Each feature vector x_i is treated independently and used to generate the block membership b_i. Each entry $b_i \in \{1, 2 \dots B\}$ where B is the number of blocks and a constant in our model. We choose a single softmax layer to model $p(b_i|x_i, \theta)$. More complex models are possible but then deriving meaning from inferred parameter distributions is more difficult. Summarising, we write $p(b|X, \theta)$ as follows:

$$p(b|X,\theta) = \prod_{i=1}^{N} p(b_i|x_i,\theta) = \prod_{i=1}^{N} \phi_{b_i}(x_i;\theta) = \prod_{i=1}^{N} \frac{\exp\left(w_{b_i}^T \tilde{x}_i\right)}{\sum_{k=1}^{B} \exp\left(w_k^T \tilde{x}_i\right)} \tag{3}$$

Where $\tilde{x} \coloneqq [x_1, x_2, \dots x_D, 1]^T$ is an augmented version of x that allows for a bias term. The parameters for this stage are denoted θ and consist of the all B weight vectors $\theta = \{w_k\}_{k=1}^B$. Each w_k has dimension $D+1$. We could instead write the parameters θ as a $B \times (D+1)$ matrix of weights W; this form has use computationally as then $Z_i = W\tilde{x}_i$ is the input to the softmax activation function.

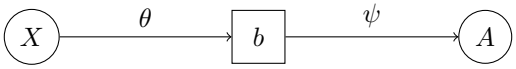


Figure 1: Latent block generative model

With the block memberships b generated, we then draw the graph A from the microcanical DC-SBM (2.2) with additional parameters $\psi = \{e, k\}$. In a slight abuse of notation we denote the inter-block edge count matrix with $e = \psi_e$ and the degree sequence $k = \psi_k$ to make explicit that these parameters are contained in ψ.

$$A \sim \text{DC-SBM}_{\text{MC}}(b, \psi_e, \psi_k) \tag{4}$$

3.1 Prior selection

Before performing any inference, we must specify priors on θ and ψ. For θ it seems sensible to choose a Gaussian prior, with zero mean and variance matrix $\sigma_\theta^2 I$ such that each element of θ is independent and distributed like $\sim \mathcal{N}(0, \sigma_\theta^2)$. In vector form, the prior for θ is therefore:

$$p(\theta) = \mathcal{N}\left(\theta; 0, \sigma_\theta^2 I\right) \tag{5}$$

We will see that this form of prior is equivalent to a regularisation term in neural network training that penalises extreme weight magnitudes. For $\sigma_\theta^2 \to \infty$ this becomes an uninformative uniform prior.

In our model, the block memberships vector b is an intermediate latent variable and so we are not free to choose a prior for it. Nevertheless, as far as inference on the right-hand-side of figure 1, we regard $p(b|X)$ as a pseudo-prior on b. We can show (appendix A.1) that our choice of prior for $p(\theta)$ in equation 5 leads to a uniform $p(b|X)$ in equation 6.

$$p(b|X) = \int p(b|X,\theta)p(\theta)d\theta = B^{-N} \tag{6}$$

This is a uniform distribution that only depends on the number of blocks B. This simplifies the problem nicely as evaluating $p(b|X)$ does not require knowing the exact value X takes. Peixoto

[15] proposes careful choices for the additional microcanonical SBM parameters ψ which we adopt. Peixoto's idea is to write the joint prior on (b, e, k) as a product of conditionals $p(b, e, k) = p(b)p(e|b)p(k|e, b) = p(b)p(\psi|b)$. For our purposes we must insert a conditioning on X, to form our pseudo-prior for b and ψ, to give equation 7.

$$p(b, \psi|X) = p(b|X)p(\psi|b, X) = p(b|X)p(\psi|b) \tag{7}$$

Where the simplification is made apparent by noting $(\psi \perp\!\!\!\perp X)|b$. We then borrow the priors proposed by Peixoto [15] for $p(\psi|b)$, repeated here for reference.

$$p(\psi|b) = p(e|b)p(k|e, b) = \left[\left\{\!\!\left\{ \begin{matrix} B \\ 2 \end{matrix} \right\}\!\!\right\} \right]^{-1} \cdot \left[\prod_r \frac{\prod_j \eta_j^r!}{n_r! q(e_r, n_r)}\right] \tag{8}$$

Where $\left\{\!\!\left\{ \begin{matrix} n \\ m \end{matrix} \right\}\!\!\right\}$ is shorthand for $\binom{n+m-1}{m} = \frac{(n+m-1)!}{(n-1)!(m)!}$ which can be thought of as the total number of distinct histograms with n bins under the constraint they sum to m. $E = \frac{1}{2}\sum_{r,s} e_{rs}$ is the total number of edges in the graph. Importantly, E is not allowed to vary and so $p(e|b)$ is uniform with respect to e. The variable η_j^r is introduced to denote the number of vertices in block r that have degree j. Formally, $\eta_j^r \coloneqq \sum_i \mathbb{1}\{b_i = r\}\,\mathbb{1}\{k_i = j\}$. Furthermore, $q(m, n)$ is the number of different histograms with at most n non-zero bins that sum to m. $q(m, n)$ is related to but different from $\left\{\!\!\left\{ \begin{matrix} n \\ m \end{matrix} \right\}\!\!\right\}$. Lastly, $e_r \coloneqq \sum_s e_{rs}$ is the total number of half edges in block r and $n_r \coloneqq \sum_i \mathbb{1}\{b_i = r\}$ is the number of vertices assigned to block r. Importantly, we have computable forms for $p(\theta)$ and $p(b, \psi|X)$ which will be useful for performing inference.

4 Inference

Now that we have defined the FFBM, we wish to leverage it to perform inference. Suppose we are presented with a vertex-labelled graph (A, X); the goal is to draw samples for θ according to the posterior given the observed graph (equation 9).

$$\theta^{(t)} \sim p(\theta|A, X) \tag{9}$$

These samples allow us to approximate the posterior distribution for θ as well as compute a predictive distribution $p(b^*|x^*, A, X) = \int p(b^*|x^*, \theta)p(\theta|X, A)d\theta \approx \frac{1}{T}\sum_{t=1}^{T} p(b^*|x^*, \theta^{(t)})$. However, generating these samples is not easily done in practice.

We instead propose an iterative approach. First drawing samples $b^{(t)}$ from the block membership posterior (equation 10). We then use each $b^{(i)}$ to draw samples for θ as in equation 11.

$$b^{(i)} \sim p\Big(b|A, X\Big) \tag{10}$$

$$\theta^{(i)} \sim p\Big(\theta|X, b^{(i)}\Big) \tag{11}$$

Both of these sampling steps implemented with a Markov Chain through the Metropolis-Hastings algorithm [5]. We just need to define a proposal distribution $q(x, x')$ for proposing a move $x \to x'$ and be able to evaluate an un-normalised form of the target distribution, denoted $\pi(\cdot)$, point-wise. The proposed move is then accepted with probability α (equation 12) else it is rejected and we stay at x.

$$\alpha = \min\left(\frac{\pi(x')q(x', x)}{\pi(x)q(x, x')}, 1\right) \tag{12}$$

This accept-reject step ensures the resulting Markov Chain is in detailed balance with the target distribution $\pi(\cdot)$. What we propose in equations 10 and 11 is therefore implemented through a 2-level Markov chain. The resulting samples for $\theta^{(i)}$ are unbiased in the sense that the expectation of their distribution is the posterior we are targeting in equation 9.

$$\begin{aligned}
\mathbb{E}_{b^{(t)}}\left[p\left(\theta|X, b^{(t)}\right)\right] &= \sum_{b \in \mathcal{B}^N} p(\theta|X, b)p(b|A, X) \\
&= \sum_{b \in \mathcal{B}^N} p(\theta, b|A, X) \\
&= p(\theta|A, X)
\end{aligned}$$

This is an example of a pseudo-marginal approach [3]. Indeed, the unbiased result is sufficient to prove that for sufficient samples, $\theta^{(t)} \sim \mathbb{E}_{b^{(i)}}\left[p(\theta|X, b^t)\right] = p(\theta\ A, X)$ which is exactly the distribution we were targetting (equation 9).

The reason we split the Markov chain into two stages is because the summation over all latent states $b \in \mathcal{B}^N$ required to directly compute the likelihood $p(A|X, \theta) = \sum_{b\in\mathcal{B}^N} p(A|b)P(b|X, \theta)$ is intractable $O(B^N)$. Figure 2 shows an overview of the proposed method. We have introduced

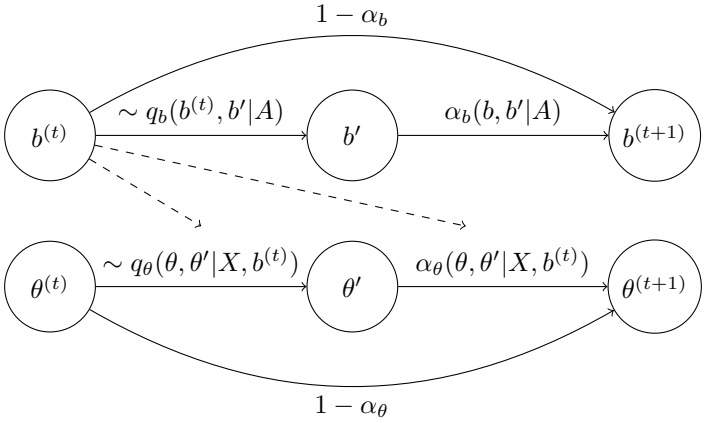


Figure 2: Sampling sequence

subscripts and conditionings to make explicit what parameters each step utilises. In an important simplification, we note that $p(b|X) = B^{-N}$ which does not depend on the exact value of X. Therefore, we do not need to know the value of X to perform the sampling on b. Conversely, for the $\theta^{(t)}$ samples, we use $b^{(t)}$ but not A as $(\theta \perp\!\!\!\perp A)|b$.

4.1 Sampling block memberships

Peixoto [13] proposes a Monte Carlo method which we will base our approach on. It relies on writing the posterior in the following form:

$$p(b|A, X) \propto p(A|b, X) \cdot p(b|X) = \pi_b(b) \tag{13}$$

Now $\pi_b(\cdot)$ is the un-normalised density we wish to sample from. In other words, we wish to construct a Markov chain that has $\pi_b(\cdot)$ as its invariant distribution. We can break π_b down as follows:

$$\begin{aligned}\pi_b(b) &= p(b|X)\sum\nolimits_\psi p(A, \psi|b, X)\\ &= p(b|X)p(A, \psi^*|b, X)\\ &= p(A|b, \psi^*) \cdot p(\psi^*|b) \cdot p(b|X)\end{aligned}$$

Since we are using the microcanonical SBM formulation, there is only one value of ψ that is compatible with the given (A, b) pair. We denote this value $\psi^* = \{k^*, e^*\}$. Specifically, $k_i^* = \sum_j A_{ij}$ and $e_{rs}^* = \sum_{i,j} A_{ij}\mathbb{1}\{b_i = r\}\mathbb{1}\{b_j = s\}$. Therefore, the summation over all ψ reduces to just the single ψ^* term. We also define the microcanonical entropy of the configuration as.

$$S(b) = -\log \pi_b(b) = -\Big(\log p(A|b, \psi^*) + \log p(\psi^*, b|X)\Big) \tag{14}$$

This entropy can be thought of as the description length of the graph because it is the sum of the information required to represent the graph given the parameters and the amount of information required to store the parameters (given the feature matrix X). The exact, from of the proposal distribution and accept-reject step is explored throughly by Peixoto [13]. There is a widely used library for Python made available under LGPL called `graph-tool` [14], which implements this algorithm. The only modification we make is in the block membership prior $p(b)$ which we replace with $p(b|X) = B^{-N}$ which is a uniform distribution and so cancels out in the MH accept-reject step.

The end result of this is that we can generate a set of block membership samples $\{b^{(t)}\}_{t=1}^T$ with each $b^{(t)} \sim p(b|A, X)$. Each of these samples can then be used for the θ-chain.

4.2 Sampling feature-to-block classifier parameters

The invariant distribution we wish to target for the θ samples is the posterior of θ given the values of the pair (X, b). We write this as follows:

$$p(\theta|X, b) \propto p(b|X, \theta)p(\theta) = \pi_\theta(\theta) \propto \exp\left(-U(\theta)\right) \tag{15}$$

$$\therefore U(\theta) = -\left(\log p(b|X, \theta) + \log p(\theta)\right) + \text{const} \tag{16}$$

Where we have introduced $U(\theta)$ equal to the negative log posterior. This $U(\theta)$ term makes subsequent analysis more concise. Each of the constituent terms of $U(\theta)$ is easily computed (equation 17). To simplify notation, we define $y_{ij} \coloneqq \mathbb{1}\{b_i = j\}$ and $a_{ij} \coloneqq \phi_j(x_i; \theta)$.

$$\log p(b|X, \theta) = \sum_{i=1}^{N}\sum_{j=1}^{B} y_{ij} \log a_{ij} \quad \text{and} \quad \log p(\theta) = -\frac{(D+1)(B)}{2}\log 2\pi - \frac{1}{2\sigma_\theta^2}||\theta||^2 \tag{17}$$

This means that if we ignore constant terms we can write $U(\theta)$ as in equation 18. We see that the prior effectively introduces a regularisation term. Note that $||\theta||^2 = \sum_i \theta_i^2 = \sum_{j=1}^{B} ||w_j||^2$ is the Euclidean norm of the vector of parameters θ.

$$U(\theta) = \left(\sum_{i=1}^{N}\sum_{j=1}^{B} y_{ij} \log \frac{1}{a_{ij}}\right) + \frac{1}{2\sigma_\theta^2}||\theta||^2 = N \cdot \mathcal{L}(\theta) + \frac{1}{2\sigma_\theta^2}||\theta||^2 \tag{18}$$

$U(\theta)$ in equation 18 appears a typical objective function to be minimised for neural network training. The first term – which we collect into $N \cdot \mathcal{L}(\theta)$ – is the cross-entropy between graph-predicted and feature-predicted block memberships. We denote the average cross-entropy $\mathcal{L}(\theta)$ as it will become useful later on when performing experiments. The second term of equation 18 – introduced by the prior – brings a form of regularisation, preventing over-fitting. In traditional applications we only seek the value of θ that minimises the objective function $U(\theta)$, which in our case would yield the maximum a posteriori (MAP) estimate. This is often done through some kind of gradient descent as ∇U is easily computable (equation 21).

However, our goal is not to find the MAP estimate but to draw samples from the posterior $\pi_\theta(\cdot) \propto \exp(-U(\cdot))$. As discussed earlier, given the invariant $\pi_\theta(\cdot)$, it is sufficient to specify a proposal distribution and then apply a MH accept-reject step to ensure detailed balance of the Markov Chain. Nevertheless, we can use ∇U as a useful heuristic to bias our proposal towards regions of higher target density. We therefore adopt the Metropolis Adjusted Langevin Algorithm (MALA) – first proposed by Roberts and Tweedie [16] – which leverages just that. Given the current sample θ, we propose a new sample θ' according to equation 19

$$\theta' = \theta - h\nabla U(\theta) + \sqrt{2h} \cdot \xi \tag{19}$$

Where $\xi \sim \mathcal{N}(0, I)$ and h is a step-size parameter - which may vary with the sample index. Indeed without the injected noise term, this is equivalent to gradient descent. We require the noise term to fully explore the parameter space. As such the proposal distribution, is a simple multivariate Gaussian which can be easily evaluated.

$$q_\theta(\theta, \theta') = \mathcal{N}\left(\theta'; \theta - h\nabla U(\theta), 2hI\right) \tag{20}$$

The term ∇U has an easy to compute analytic form (derived in Appendix A.2). By noting that $\theta = \{w_k\}_{k=1}^{B}$, we write the derivative with respect to each w_k as:

$$\frac{\partial U}{\partial w_k} = -\left(\sum_{i=1}^{N}\left\{\tilde{x}_i(y_{ik} - a_{ik})\right\} - \frac{w_k}{\sigma_\theta^2}\right) \tag{21}$$

After a proposed move is generated, in typical Metropolis-Hastings fashion we accept the move with probability α_θ.

$$\alpha_\theta(\theta, \theta') = \min\left(\exp\left(U(\theta) - U(\theta')\right)\frac{q_\theta(\theta', \theta)}{q_\theta(\theta, \theta')}, 1\right) \tag{22}$$

This fully specifies, the sampling procedure to generate $\{\theta^{(t)}\}_{t=1}^{T}$. Bear in mind that each $\theta^{(t)}$ update step uses its corresponding $b^{(t)}$ block membership sample.

4.3 Sampling sequence

So far, each $\theta^{(t)}$ update uses its corresponding $b^{(t)}$ sample. This means that the evaluation of $U(\theta)$ and $\nabla U(\theta)$ has high variance. This may lead to longer burn-in and autocorrelation times of the resulting Markov Chain. The only link between $b^{(t)}$ and $\theta^{(t)}$ is in the evaluation of $U(\theta)$ and $\nabla U(\theta)$ which depends on $y_{ij}^{(t)} \coloneqq \mathbb{1}\{b_i^{(t)} = j\}$. We would rather deal with the expectation of each $y_{ij}^{(t)}$:

$$\mathbb{E}\left[y_{ij}^{(t)}\right] = \mathbb{E}_{b^{(t)}}\left[\mathbb{1}(b_i^{(t)} = j)\right] = p(b_i = j|A, X) \tag{23}$$

We obtain an unbiased estimate for this quantity as simply the empirical distribution of the block membership samples $\left\{b^{(t)}\right\}_{t=1}^{T}$.

$$\hat{y}_{ij} \coloneqq \frac{1}{T}\sum_{t=1}^{T} y_{ij}^{(t)} = \frac{1}{T}\sum_{t=1}^{T} \mathbb{1}\{b_i^{(t)} = j\} \tag{24}$$

We therefore, choose to feed each $\theta^{(t)}$ update step the same $\hat{y}_{ij}$ for all t rather than the corresponding $y_{ij}^{(t)}$. This means we no longer need to run the b and θ Markov chains concurrently. Instead, we run the b-chain to completion and use it to generate $\hat{y}_{ij}$ for $i \in \{1 \dots N\}$ and $j \in \{1 \dots B\}$. This is an estimate of $p(b|A, X)$ that we use for every iteration of the θ Markov chain. This affords us the flexibility to vary the number of samples we draw for b and θ; we refer to these as T_b and T_θ henceforth. Furthermore, this changeover reduces the burn-in time for the θ-chain by reducing the variance in our evaluation of U and ∇U.

TODO: burn-in and thinning

4.4 Dimensionality reduction

Once we have the samples $\left\{\theta^{(t)}\right\} \sim p(\theta|A, X)$, we can compute the empirical mean and standard deviation of each component of θ. Switching back to matrix notation we define $\theta = W$, such that W_{ij} is the weight component for block i and feature j, we can define:

$$\hat{\mu}_{ij} \coloneqq \frac{1}{|\mathcal{T}_\theta|}\sum_{t\in\mathcal{T}_\theta} W_{ij}^{(t)} \qquad \text{and} \qquad \hat{\sigma}_{ij} \coloneqq \frac{1}{|\mathcal{T}_\theta|}\sum_{t\in\mathcal{T}_\theta}\left(W_{ij}^{(t)} - \hat{\mu}_{ij}\right)^2 \tag{25}$$

A simple heuristic to discard the least important features requires specifying a cutoff $c > 0$ and a multiplier $k > 0$. We define the function $\mathcal{F}_i(j)$ as in 26 then only keep features in $\mathcal{D}'$ given by equation 27.

$$\mathcal{F}_i(j) \coloneqq (\hat{\mu}_{ij} - k\hat{\sigma}_{ij}, \hat{\mu}_{ij} + k\hat{\sigma}_{ij}) \cap (-c, +c) \tag{26}$$

$$\mathcal{D}' = \{j \in [D] : \exists i \in [B] \text{ s.t. } \mathcal{F}_i(j) \neq \emptyset\} \tag{27}$$

Intuitively, this means discarding any feature for which $\hat{\mu}_{ij} \pm k\hat{\sigma}_{ij}$ lies within or spans the null region $(-c, c)$ for all block indices. If we were to use the Laplace approximation for the posterior $p(W_{ij}|A, X) \approx \mathcal{N}(W_{ij}; \mu_{ij}, \sigma_{ij})$, then this is effectively a hypothesis test on the value of W_{ij}. $\mathcal{D}'$ then comprises all features for which H_1 is accepted.

$$\begin{aligned} H_0 &: |W_{ij}| \leq c \\ H_1 &: |W_{ij}| > c \end{aligned} \tag{28}$$

The multiplier k specifies the degree of significance of the result (equivalent to the probability of a type-1 error). However, as the approximation is not exact we will only treat this dimensionality reduction method as a useful heuristic and not an exact method.

Conversely, we could fix $k = k_0$ and the dimension of our reduced feature set $|\mathcal{D}'| = D'$. We would then like to find the largest value of c such that $|\mathcal{D}'| = D'$ given $k = k_0$. This is summarised in equation 29. This approach may be preferred as it fixes the number of reduced dimensions.

$$c^* = \arg\max_{c>0}(c : |\mathcal{D}'| = D', k = k_0) \tag{29}$$

5 Experiments

We apply the developed methods to a variety of datasets, to show the power of the method across different scenarios. We choose datasets to span a range of node counts N, edge counts E and feature space dimension D. We consider the following datasets:

- **Political books** [11] ($N = 105, E = 441, D = 3$) - network of Amazon book sales about U.S. politics, published close to the presidential election in 2004. Two books are connected if they were frequently co-purchased by the same customer. Vertex features encode the broad political affiliation of the author (liberal, conservative or neutral).
- **Primary school dynamic contacts** [18] ($N = 238, E = 5539, D = 13$) - network of face-to-face contacts amongst students and teachers at a primary school in Lyon, France. Two nodes are connected if the two parties shared a face-to-face interaction over the course of the day. Vertex features include class membership, gender and whether or not the individual is a teacher or pupil. These data were collected on consecutive days in October 2009. We choose to analyse just the second day.
- **Facebook egonet** [7] ($N = 747, E = 30025, D = 480$) - an assortment of Facebook egonets. These are networks of a particular user's friends list and all the connections within that. Vertex features are extracted from each user's profile and are fully anonymized. We focus on the egonet with id 1912.

We require metrics to assess the relative model evidence for the FFBM. This can be split into two separate components: the microcanonical SBM fit (concerned with the b-samples) and the fit of the feature-to-block generator (concerned with the θ-samples). The SBM has been extensively analysed in other works [15] so that will not be the emphasis of our analysis. $S(b)$ (equation 14) can be interpreted as the description length of the partition imposed by b. In other words this is the amount of information needed to describe the graph and the associated SBM parameters. It is only natural to divide this quantity by the number of entities (nodes and edges) in our graph $N + E$ to allow for rough comparison between graphs. We use a simple metric to gauge the fit of the SBM: the description length per entity averaged over the b-samples (equation 30):

$$\bar{S}_e \coloneqq \frac{1}{(N+E)|\mathcal{T}_b|} \sum_{t \in \mathcal{T}_b} S\left(b^{(t)}\right) \tag{30}$$

However, to assess the performance of the feature-to-block predictor, we must partition the vertex set $[N]$ into training and test sets – $\mathcal{G}_0$ and $\mathcal{G}_1$ respectively [1] . The b-chain is run using the whole network but we only use vertices $v \in \mathcal{G}$ to train the θ-chain. As $|\mathcal{G}_0| \neq |\mathcal{G}_1|$ sets, we cannot use the un-normalised log target U (equation 18) for comparison between the two, as the prior term will be equal between the two but the total cross-entropy loss will be scaled by the size of each set. We therefore must use the average cross-entropy loss over each set (equation 31).

$$\bar{\mathcal{L}}_a \coloneqq \frac{1}{|\mathcal{T}_\theta|} \sum_{t \in \mathcal{T}_\theta} \mathcal{L}_a\left(\theta^{(t)}\right) \qquad \text{where} \qquad \mathcal{L}_a\left(\theta^{(t)}\right) \coloneqq \frac{1}{|\mathcal{G}_a|} \sum_{i \in \mathcal{G}_a} \sum_{j \in [B]} \hat{y}_{ij} \log \frac{1}{\phi_j\left(x_i; \theta^{(t)}\right)} \tag{31}$$

Where $a \in \{0, 1\}$ has been introduced to toggle between training and test sets. Table 1 summarises the results for each experiment. We run each experiment 5 times with the same hyperparameter values[2] and report the mean plus or minus one standard deviation.

However, we also wish to determine the power of the dimensionality reduction method. For this we discard all features that satisfy equation **??**, to reduce the dimension from D to D'. We then retrain the feature-block predictor using just the retained features and report the loss over the training and test sets for the reduced classifier – denoted $\bar{\mathcal{L}}'_0$ and $\bar{\mathcal{L}}'_1$ respectively. These values are also included in the table except for the polbooks dataset as this is already very low.

[1] We choose to randomly create the partition on each experiment run such that a fraction f of the available vertices form our training set $\mathcal{G}$ and the remaining vertices are held out to form our test set $\mathcal{G}^C$. For all the experiments $f = 0.7$.

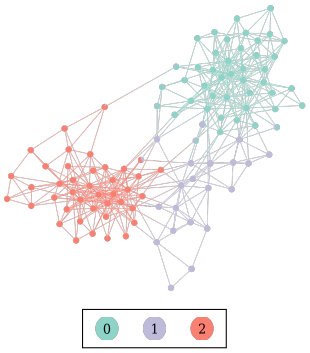
[2] TODO hyperparam values

Table 1: Results

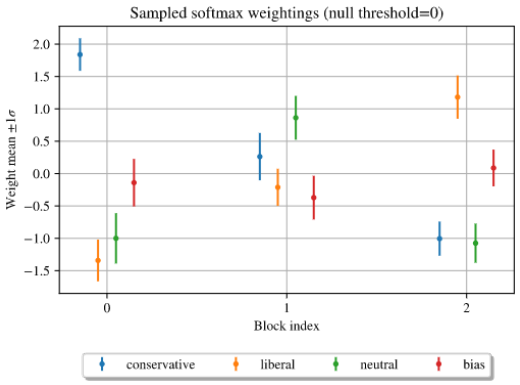
Dataset	B	D	D'	$\bar{S}_e$	$\bar{\mathcal{L}}_0$	$\bar{\mathcal{L}}_1$	c^*	$\bar{\mathcal{L}}'_0$	$\bar{\mathcal{L}}'_1$
Polbooks	3	3	–	2.250 ± 0.001	0.584 ± 0.033	0.557 ± 0.061	–	–	–
School	10	13	10	1.895 ± 0.006	0.789 ± 0.109	0.876 ± 0.135	1.114 ± 0.224	0.786 ± 0.103	0.833 ± 0.126
FB egonet	10	480	10						

5.1 Political books

This dataset was collected by Pasternak and Ivask [11]. We wish to determine whether political affiliation is a good predictor of the overall network structure. We choose to partition the network into $B = 3$ communities as we only have this many distinct values for political affiliation (conservative, liberal or neutral). The inferred block memberships are given in figure 3a. We sample the block-generator parameters and plot the emprical mean and standard deviation of each sample on figure 3b. Indeed, for all 3 blocks, each has a distinct political affiliation as its highest magnitude component.



(a) Sampled block memberships $\hat{Y}$



(b) Feature weights

Figure 3: Political books network [11]

This is strong evidence that political affiliation is indeed the axis which best predicts the 3-way natural partition of the graph into blocks. Nevertheless, for block index 1, although `neutral` is the best predictor of the ones we have available, the value is not particularly extreme. Perhaps, there is an unobserved feature that places a book into this block. Indeed, block-0 is more of a centre-right block than true centre.

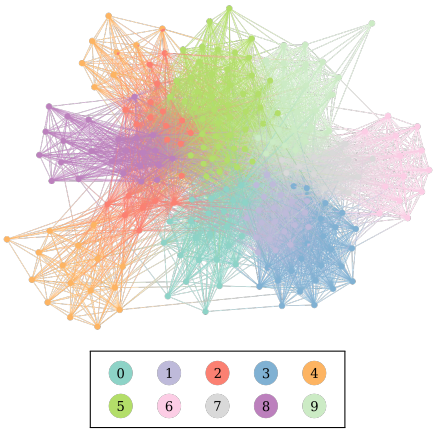
5.2 Primary school dynamic contacts

These data were originally collected by Stehlé et al. [18] to quantify the transmission opportunities for respiratory infections. However, we seek to ask which vertex features best describe how people interact with one another in a primary school context. The only vertex features we have available are school-class (one of 10 values - 2 per year group), gender and a distinction between teachers and pupils.

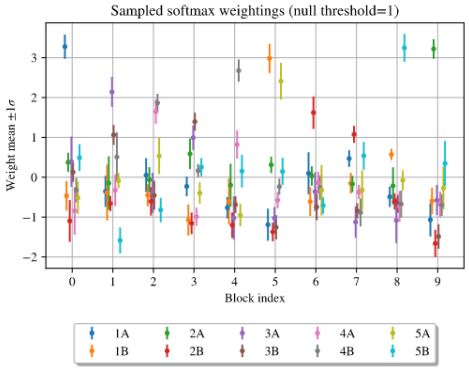
We must first choose the number of blocks B to define the coarseness of our analysis. A total of 10 school-classes would suggest that $B = 10$ is a natural starting point. We visualise the inferred block memberships in figure 4a.

As before, we proceed to sample the block-generator parameters θ and employ the dimensionality reduction heuristic defined in section 4.4 with null threshold $c = 1$ and standard deviation multiplier $k = 1$. We then plot the weights for the features that survive the discard step in figure 4.4. Immediately, we see that only the pupils' class memberships have survived (1A-5B); gender and teacher/student status have been discarded meaning that these are not good predictors of overall macro-structure.

The vast majority of blocks are composed of a single class. However, some blocks have 2 comparably good classes as their predictor. For example, blocks 2 and 4 contain classes 4A and 4B as their 2 best predictors. This suggests that the social divide between classes is less pronounced for pupils in year 4 but there is nonetheless an unobserved variable that splits year 4 into 2 communities. The most



(a) Sampled block memberships $\hat{Y}$



(b) Reduced dimension feature weights

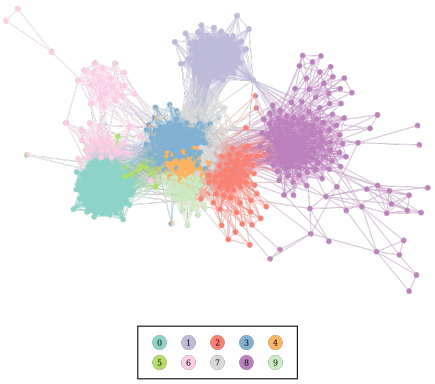
Figure 4: Primary school dynamic contacts network [18]

surprising block is number 5 - which has comparable weightings for classes 5A and 1B. Perhaps there was a joint event between those two classes on the day the data were collected.

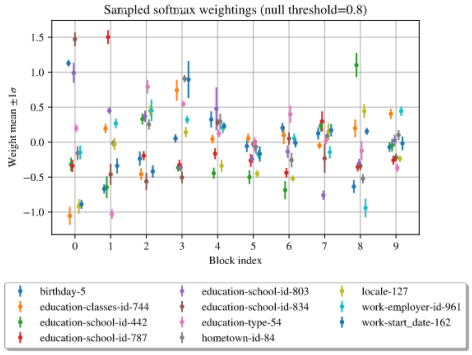
5.3 Facebook egonet

This dataset was chosen to showcase the power of the dimensionality reduction technique as the feature-space has high dimension ($D = 480$). We sample the b-chain specifying $B = 10$ total blocks and use this to construct the θ-samples as before.

We apply the dimensionality reduction heuristic outlined in section 4.4, choosing $k = 1$ and $c = 0.8$. This leads to a reduced dimension $D' = 13$. These parameter values are plotted on figure 5b. The features that remain are those that best explain the high-level community structure. More so than language, gender or surname - the school[3] you attend is the best explanatory variable for how the observed network's communities are partitioned at the highest level.



(a) Sampled block memberships $\hat{Y}$



(b) Reduced dimension feature weights

Figure 5: Facebook egonet id 1912 [7]

Nevertheless, this example also highlights a weakness of the method for high-dimensional feature spaces. When the feature dimension is very large, it becomes increasingly likely that a particular feature may uniquely identify a small set of nodes. If these nodes are all part of the same community then the classifier will overfit for that particular parameter. The regularisation term imposed by the

[3]This includes higher education institutes

prior goes some way to alleviating this problem but we see in figure 5b that the feature `birthday-5` has a very high weight as it relates to block 0. It might be possible to shift the feature values such that they take values in $\{-1, 1\}$ rather than $\{0, 1\}$ but this approach falls down when input features are mutually correlated. Indeed, for features with discrete values such as the class memberships (1A, 1B, 2A ... 5B) it is preferred to keep $x_i \in \{0, 1\}^D$ as then each block can be readily accept more than one mutually exclusive feature-group (i.e. two school-classes are part of the same block).

6 Conclusion

The FFBM was developed to address the shortcomings of other graphical models when testing how vertex features affect community structure. The idea is to divide the graph into its most natural partition and only then bring in the vertex features. We therefore hope to avoid a common pitfall in data science: correlation does not imply causation. It is very easy to find vertex features that are in some way correlated with the graphical structure. Nonetheless, only when we find the feature that best describes the most pronounced partition do we have a stronger case for causation.

With the newly-defined FFBM, we go on to present an efficient inference algorithm to ascertain the parameters θ of the block generator. This is introduced as two concurrent Markov chains to sample the block memberships b and block generator parameters θ. The key trick to make the b-chain efficient is the the use of the microcanonical rather than the traditional SBM (as proposed by Peixoto [15]) which allows us to bypass an expensive summation over the latent memberships b. Additionally, we use the empirical mean of the b-samples as the input to our θ-chain, as this reduces the variance in our evaluation of the target distribution and thus shortens the burn-in and auto-correlation times of our θ-chain.

The overall method is shown to be effective at extracting and describing the most natural communities in a labelled network. Nevertheless, the approach can only currently explain the structure at the macro-scale. We cannot explain structure within each community. Future work will benefit from extending the FFBM to be hierarchical in nature. That way, the structure of the network can be explained at all length-scales of interest.

Acknowledgments and Disclosure of Funding

Unsure what to put in this section

Use unnumbered first level headings for the acknowledgments. All acknowledgments go at the end of the paper before the list of references. Moreover, you are required to declare funding (financial activities supporting the submitted work) and competing interests (related financial activities outside the submitted work). More information about this disclosure can be found at: `https://neurips.cc/Conferences/2021/PaperInformation/FundingDisclosure`.

Do **not** include this section in the anonymized submission, only in the final paper. You can use the `ack` environment provided in the style file to automatically hide this section in the anonymized submission.

References

[1] Emmanuel Abbe. Graph compression: The effect of clusters. In *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1–8, 2016. doi: 10.1109/ALLERTON.2016.7852203.

[2] Edo M Airoldi, David Blei, Stephen Fienberg, and Eric Xing. Mixed membership stochastic blockmodels. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2009. URL `https://proceedings.neurips.cc/paper/2008/file/8613985ec49eb8f757ae6439e879bb2a-Paper.pdf`.

[3] Christophe Andrieu and Gareth O. Roberts. The pseudo-marginal approach for efficient Monte Carlo computations. *The Annals of Statistics*, 37(2):697 – 725, 2009. doi: 10.1214/07-AOS574. URL `https://doi.org/10.1214/07-AOS574`.

[4] Solenne Gaucher, Olga Klopp, and Geneviève Robin. Outliers detection in networks with missing links, 2020.

[5] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970. ISSN 00063444. URL `http://www.jstor.org/stable/2334940`.

[6] Matthew Kraatz, Nina Shah, and Emmanuel Lazega. The collegial phenomenon: The social mechanisms of cooperation among peers in a corporate law partnership. *Administrative Science Quarterly*, 48:525, 09 2003. doi: 10.2307/3556688.

[7] Jure Leskovec and Julian Mcauley. Learning to discover social circles in ego networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL `https://proceedings.neurips.cc/paper/2012/file/7a614fd06c325499f1680b9896beedeb-Paper.pdf`.

[8] Benjamin F. Maier and Dirk Brockmann. Cover time for random walks on arbitrary complex networks. *Phys. Rev. E*, 96:042307, Oct 2017. doi: 10.1103/PhysRevE.96.042307. URL `https://link.aps.org/doi/10.1103/PhysRevE.96.042307`.

[9] Nikhil Mehta, Lawrence Carin Duke, and Piyush Rai. Stochastic blockmodels meet graph neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4466–4474. PMLR, 09–15 Jun 2019. URL `http://proceedings.mlr.press/v97/mehta19a.html`.

[10] Krzysztof Nowicki and Tom A. B Snijders. Estimation and prediction for stochastic blockstructures. *Journal of the American Statistical Association*, 96(455):1077–1087, 2001. doi: 10.1198/016214501753208735. URL `https://doi.org/10.1198/016214501753208735`.

[11] Boris Pasternak and Ivor Ivask. Four unpublished letters. *Books Abroad*, 44(2):196–200, 1970. ISSN 00067431. URL `http://www.jstor.org/stable/40124305`.

[12] Tiago P. Peixoto. Parsimonious module inference in large networks. *Physical Review Letters*, 110(14), Apr 2013. ISSN 1079-7114. doi: 10.1103/physrevlett.110.148701. URL http://dx.doi.org/10.1103/PhysRevLett.110.148701.

[13] Tiago P. Peixoto. Efficient monte carlo and greedy heuristic for the inference of stochastic block models. *Physical Review E*, 89(1), Jan 2014. ISSN 1550-2376. doi: 10.1103/physreve.89.012804. URL http://dx.doi.org/10.1103/PhysRevE.89.012804.

[14] Tiago P. Peixoto. The graph-tool python library. *figshare*, 2014. doi: 10.6084/m9.figshare.1164194. URL http://figshare.com/articles/graph_tool/1164194.

[15] Tiago P. Peixoto. Nonparametric bayesian inference of the microcanonical stochastic block model. *Physical Review E*, 95(1), Jan 2017. ISSN 2470-0053. doi: 10.1103/physreve.95.012317. URL http://dx.doi.org/10.1103/PhysRevE.95.012317.

[16] Gareth O. Roberts and Richard L. Tweedie. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341 – 363, 1996. doi: bj/1178291835. URL https://doi.org/.

[17] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding, 2019.

[18] Juliette Stehlé, Nicolas Voirin, Alain Barrat, Ciro Cattuto, Lorenzo Isella, Jean-François Pinton, Marco Quaggiotto, Wouter Van den Broeck, Corinne Régis, Bruno Lina, and Philippe Vanhems. High-resolution measurements of face-to-face contact patterns in a primary school. *PLOS ONE*, 6(8):1–13, 08 2011. doi: 10.1371/journal.pone.0023176. URL https://doi.org/10.1371/journal.pone.0023176.

[19] Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 681–688, Madison, WI, USA, 2011. Omnipress. ISBN 9781450306195.

[20] Jun Zhu, Jiaming Song, and Bei Chen. Max-margin nonparametric latent feature models for link prediction, 2016.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section gen-inst.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See section conclusion
   (c) Did you discuss any potential negative societal impacts of your work? **[TODO]**
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? **[TODO]**
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See section 4
   (b) Did you include complete proofs of all theoretical results? [Yes] See section 4
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? **[TODO]**
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? **[TODO]**
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? **[TODO]**
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? **[TODO]**
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? **[TODO]**
   (b) Did you mention the license of the assets? **[TODO]**
   (c) Did you include any new assets either in the supplemental material or as a URL? **[TODO]**
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? **[TODO]**
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? **[TODO]**
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

A.1 Derivation of conditional block distribution given feature matrix

We wish to determine the form of $p(b|X)$. This can be done by integrating over the joint probability with respect to θ.

$$\begin{aligned} p(b|X) &= \int p(b,\theta|X,\theta)d\theta = \int p(b|X,\theta)p(\theta|X)d\theta \\ &= \int p(b|X,\theta)p(\theta)d\theta = \int \prod_{i=1}^{N} \phi_{b_i}(x_i;\theta)p(\theta)d\theta \\ &= \prod_{i=1}^{N} \int \frac{\exp(w_{b_i}^T \tilde{x}_i) \prod_{j=1}^{B} \mathcal{N}(w_j; 0, \sigma_\theta^2 I)}{\sum_{k=1}^{B} \exp(w_k^T \tilde{x}_i)} dw_{1:B} \end{aligned}$$

We note that $b_i \in 1, 2, \dots B$ and so the integral's value is unchanged with respect to b_i. The integrand has the same form no matter which value b_i takes as the prior is the same for each w_j. As such the integral can only be a function of at most $\tilde{x}_i$ and σ_θ^2 as it is symmetric with respect to b_i and all the various w_j are integrated out as they are dummy variables. Therefore, denoting the integral by the (unknown) function $f(\tilde{x}_i, \sigma_\theta^2)$, we write $p(b|X)$ as follows:

$$p(b|X) = \prod_{i=1}^{N} f(\tilde{x}_i, \sigma_\theta^2) = \text{const w.r.t } b = c$$

As this is a constant with respect to b we conclude that $p(b|X)$ must be a uniform distribution. $1/c$ is simply the size of the set of values that b can take. We know $b_i \in \mathcal{B} = \{1, 2, \dots B\}$. Therefore, $b \in \mathcal{B}^N$ and $|\mathcal{B}^N| = |\mathcal{B}|^N = B^N = 1/c$. Putting this all together we show that:

$$p(b|X) = B^{-N} \tag{32}$$

A.2 Derivation of U gradient with respect to feature parameters

The goal is to determine $\nabla U(\theta)$, the gradient of the negative log posterior with respect to the parameters. We repeat the form of $U(\theta)$ in equation 33.

$$U(\theta) = \left(\sum_{i=1}^{N} \sum_{j=1}^{B} y_{ij} \log \frac{1}{a_{ij}} \right) + \frac{1}{2\sigma_\theta^2} ||\theta||^2 \tag{33}$$

Where y_{ij} is independent of θ and a_{ij} is the output from the softmax layer, with form as given in equation 34.

$$a_{ij} \coloneqq \phi_j(x_i;\theta) = \frac{\exp(w_j^T \tilde{x}_i)}{\sum_{b=1}^{B} \exp(w_b^T \tilde{x}_i)} \tag{34}$$

We note that $\theta = \{w_k\}_{k=1}^{B}$, and as such we can write this in vector form $\theta = \left[w_1^T, w_2^T \dots w_B^T\right]^T$. Therefore, $\nabla U(\theta) = \left[{}^{\partial U}/_{\partial w_1}{}^T, {}^{\partial U}/_{\partial w_2}{}^T \dots {}^{\partial U}/_{\partial w_B}{}^T\right]^T$; to compute $\nabla U(\theta)$ it suffices to find the form of ${}^{\partial U}/_{\partial w_k}$ with respect to a general k.

To this end, we must first find partial derivatives of a_{ij} and $||\theta||$ with respect to w_k. Starting with a_{ij}:

$$\begin{aligned} \frac{\partial a_{ij}}{\partial w_k} &= \frac{\tilde{x}_i \exp(w_j^T \tilde{x}_i)\delta_{jk} \cdot \sum_{b=1}^{B} \exp(w_b^T \tilde{x}_i) - \exp(w_j^T \tilde{x}_i) \cdot \tilde{x}_i \exp(w_k^T \tilde{x}_i)}{\left(\sum_{b=1}^{B} \exp(w_b^T \tilde{x}_i)\right)^2} \\ &= \tilde{x}_i \left(a_{ij}\delta_{jk} - a_{ij}a_{ik}\right) \end{aligned} \tag{35}$$

Where $\delta_{jk} \coloneqq \mathbb{1}\{j = k\}$. Now moving onto the derivative of $||\theta||^2$:

$$\frac{\partial}{\partial w_k} ||\theta||^2 = \frac{\partial}{\partial w_k} \left(\sum_{b=1}^{B} ||w_b||^2 \right) = 2w_k \tag{36}$$

We are ready to put this all together, to find the partial derivative of $U(\theta)$ with respect to each w_k:

$$\begin{aligned}\frac{\partial U}{\partial w_k} &= \sum_{i=1}^{N}\sum_{j=1}^{B} y_{ij}\left(\frac{-\tilde{x}_i}{a_{ij}}\left(a_{ij}\delta_{jk} - a_{ij}a_{ik}\right)\right) + \frac{w_k}{\sigma_\theta^2} \\ &= -\left(\sum_{i=1}^{N}\tilde{x}_i\left(y_{ik} - a_{ik}\sum_{j=1}^{B} y_{ij}\right) - \frac{w_k}{\sigma_\theta^2}\right) \\ &= -\left(\sum_{i=1}^{N}\left\{\tilde{x}_i(y_{ik} - a_{ik})\right\} - \frac{w_k}{\sigma_\theta^2}\right) \end{aligned} \tag{37}$$

This is the required result. This form can be computed efficiently through matrix operations. The only property of y_{ij} we have used in the derivation is the sum-to-one constraint $\sum_{j=1}^{B} y_{ij} = 1$ for all i.

A.3 Choosing the MALA step-size

For sampling from the θ-chain of the block membership generator parameters, we employed the Metropolis Adjusted Langevin Algorithm (MALA). At iteration t, the proposed sample is generated by:

$$\theta' = \theta^{(t)} - h_t \nabla U(\theta^{(t)}) + \sqrt{2h_t}\cdot\xi \tag{38}$$

There are two competing objectives when choosing the step-size h_t. On the one hand, we want the step-size to be large so that we arrive at a high density region quickly. However, too large a step-size will lead to a lower acceptance ratio and thus inefficient sampling. A solution to this problem would be to slowly decrease the step-size with t - often called simulated annealing. Therefore, we still have a short burn-in time but will not bounce around the mode for large t. As well as the trivial constraint for h_t to be strictly positive and we introduce two further constraints as outlined by Welling and Teh [19]:

$$\sum_{t=1}^{\infty} h_t = \infty \qquad \text{and} \qquad \sum_{t=1}^{\infty} h_t^2 < \infty \tag{39}$$

The first constraint ensures that we have cover sufficient distance to arrive at any arbitrary point in our domain, no matter the starting point. The second constraint ensures that once we converge to the mode rather than simply bouncing around it. Welling and Teh [19] propose the following form for a polynomially decaying step-size which we adopt:

$$h_t = \alpha(\beta + t)^{-\gamma} \tag{40}$$

Where α, β, γ are hyper-parameters to be chosen. We require $\alpha, \beta > 0$ and $\gamma \in (0.5, 1]$ to satisfy equation 39. To reduce the number of hyperparameters we set these to have values given by equation 41.

$$\alpha = \frac{250 \cdot s}{N} \qquad \beta = 1000 \qquad \gamma = 0.8 \tag{41}$$

Where N is the number of data-points we are considering and now s is the only free variable which we call the step-size scaling.

A.4 Burn-in and thinning

As with any MCMC method, we must deal with the issues presented by burn-in and thinning. We have introduced the notation $\mathcal{T}_b$ and $\mathcal{T}_\theta$ to denote the set of samples we keep from the b and θ chains respectively. Note that we generate T_b and T_θ samples total. The burn-in period refers to the time taken for the Markov Chain to converge to the stationary distribution. Sample thinning is necessary to ensure that neighbouring samples satisfy independence. However, as we do not leverage the independence property this is less important in our analysis. We can write the general set $\mathcal{T}_\star$ as:

$$\mathcal{T}_\star = \{T_\star\kappa_\star + i\lambda_\star : 0 \leq i \leq \lfloor T_\star(1-\kappa_\star)/\lambda_\star \rfloor\} \tag{42}$$

Where the parameter $\kappa_\star \in (0,1)$ controls our burn-in and $\lambda_\star$ controls our thinning. $\kappa_\star$ can be determined by plotting the log-target (either $S(b^{(t)})$ or $U(\theta^{(t)})$ with respect to the epoch t. $\kappa_\star$ is then chosen to encompass the region where the log-target has roughly equilibrated. As we do not leverage sample independence $\lambda_\star$ can be chosen less rigorously. We often just use $\lambda_\star = 5$

A.5 Hyperparameter values

Table 2: Hyperparameter values for each experiment

Dataset	B	f	σ_θ	T_b	κ_b	λ_b	T_θ	κ_θ	λ_θ	s	k	D'	T'_θ	κ'_θ	λ'_θ	s'
Polbooks	3	0.8	1	1,000	0.2	5	10,000	0.2	10	0.2	–	–	–	–	–	–
School	10	0.8	1	1,000	0.2	5	10,000	0.2	10	0.2	1	10	10,000	0.2	10	0.2
FB Egonet	10	0.8	1	1,000	0.2	5	10,000	0.2	10	0.2	1	10	10,000	0.2	10	0.2

A.6 Algorithms

Algorithm 1 Block membership sample generation

for $t \in \{0, 1 \dots T_b - 1\}$ **do**
  $b' \leftarrow\sim q_b(b^{(t)}, b'|A)$
  $\log \alpha_b \leftarrow \log \alpha_b(b^{(t)}, b'|A)$
  $\eta \leftarrow\sim \text{Unif}(0, 1)$
  if $\log \eta < \log \alpha_b$ **then**
    $b^{(t+1)} \leftarrow b'$
  else
    $b^{(t+1)} \leftarrow b^{(t)}$
  end if
end for
return $\{b^{(t)}\}_{t=1}^{T_b}$

Algorithm 2 FFBM parameter pseudo-marginal inference

$\hat{Y}_{ij} \leftarrow \frac{1}{|\mathcal{S}|} \sum_{t \in \mathcal{S}} \mathbb{1}\{b_i^{(t)} = j\} \quad \forall i, j$

for $t \in \{0, 1 \dots T_\theta - 1\}$ **do**
  $\xi \leftarrow\sim \mathcal{N}(0, I)$
  $\theta' \leftarrow \theta^{(t)} - h_t \nabla U(\theta^{(t)}|X, \hat{Y}) + \sqrt{2h_t} \cdot \xi$
  $\log \alpha_\theta \leftarrow \log \alpha_\theta(\theta^{(t)}, \theta'|A, \hat{Y})$
  $\eta \leftarrow\sim \text{Unif}(0, 1)$
  if $\log \eta < \log \alpha_\theta$ **then**
    $\theta^{(t+1)} \leftarrow \theta'$
  else
    $\theta^{(t+1)} \leftarrow \theta^{(t)}$
  end if
end for
return $\{\theta^{(t)}\}_{t=1}^{T_\theta}$

A.7 Inferring the number of blocks

For the purposes of our model (the FFBM), the number of blocks B is a constant which must be specified by the data scientist. We could however, allow our choice of B to be influenced by the observed data. This places us in the domain of empirical Bayes, which must be negotiated carefully. Prior beliefs must be determined a priori else they are not prior. However, as the number of blocks only specifies the coarseness of the analysis, it is fine to allow it to vary. Indeed, Peixoto [12] shows that for a fixed average degree the maximum number of detectable blocks scales as $O(\sqrt{N})$ where N is the number of vertices.

If we allow B to vary in the b-chain (i.e. new blocks can be created and we permit empty blocks) then it can be run until a minimum description length (MDL) solution is reached. We take the number of non-empty blocks at the MDL to be our fixed block number B for subsequent analysis. Indeed, it is prudent to start our b-chain at this MDL solution as then we can neglect the burn-in time.

A.8 Implementation details

All data analysis and visualisation was implemented in Python. Full source code is available at:
`https://github.com/LozzaTray/Jormungandr`
The scripts were run using a standard PC. Specs are:

- **CPU**: Intel(R) Core(TM) i7-1065G7
- **RAM**: 8GB
- **GPU**: Intel(R) Iris(R) Plus Graphics