Inferring community characteristics in labelled networks

Anonymous Author(s)
Affiliation
Address
email

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 (or *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 – 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

An important characteristic of many real-world networks is that they exhibit strong community structure, with most nodes often belonging to a densely connected cluster. Finding ways to recover the latent communities from the observed graphs is an important task in many applications, includng graph/network compression [1] and link prediction in incomplete networks [4].

In this work, we restrict our attention to vertex-labelled networks, and we refer to the vertex labels as *features*. A common goal is to determine whether a given feature impacts graphical structure. To answer this from a Bayesian perspective requires the use of a random graph model; the standard is the stochastic block model (SBM) [8]. 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. Numerous variants of this model have been considered – the most popular ones being the

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

mixed-membership stochastic block model (MMSBM) [2] and the overlapping stochastic block model (OSBM) [17]. Effectively, these extend the model to allow each vertex to belong to multiple blocks simultaneously. However, one 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 [7] 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 first partition the graph into blocks grouped by distinct values of the feature of interest, and then use the associated model 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 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 find which of the available features – if any – best predict the resulting partition.

Thus motivated, we present a novel framework for modelling labelled networks, which we call the feature-first block model (FFBM). This is an extension of the SBM to labelled networks. We go on to present an efficient algorithm for sampling from the parameters of the feature-to-block generator, and to describe how the sampled FFBM parameters can be interpreted in order to determine which features have the largest impact on overall graphical structure.

2 Preliminaries

We first need a model for community-like structure in a network. For this we adopt the widely-used stochastic block model (SBM). 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. Specifically, we will use the microcanonical variant of the SBM, proposed by Peixoto [13]. To allow for degree-variability between members of the same block, we adopt the degree-corrected formulation (DC-SBM).

For each integer $K \geq 1$, we use the notation $[K] := \{1, 2, \ldots, K\}$.

Definition 2.1 (Microcanonical DC-SBM) *Let $N \geq 1$ denote the number of vertices in an undirected graph. The block memberships are encoded by a vector $b \in [B]^N$, where B is the number of non-empty blocks. Let $e = (e_{rs})$ be the $N \times N$ symmetric matrix of edge counts between blocks, so that e_{rs} is the number of edges from block r to block s – or twice that number if $r = s$. Let $k = (k_i)$ denote the vector of length N containing the degree sequence of the graph, with k_i being the degree of vertex i.*

The graph's adjacency matrix $A \in \{0, 1\}^{N \times N}$ is generated by placing edges uniformly at random, but respecting the constraints imposed by e, b and k (hence the qualification 'microcanonical'). Specifically, A must satisfy the following, for all $r, s \in [B]$ and all $i \in [N]$:

$$e_{rs} = \sum_{i,j \in [N]} A_{ij} \mathbb{1}\{b_i = r\} \mathbb{1}\{b_j = s\} \qquad \text{and} \qquad k_i = \sum_{j \in [N]} A_{ij}. \tag{1}$$

We use the following notation to indicate this distribution:

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

3 Feature-first block model

In this section we propose a novel generative model for labelled networks. We call this the feature-first block model (FFBM).

Let N denote the number of nodes, B the number of blocks in the graph, and $\mathcal{X}$ the set of all possible features. We define the vector $x_i \in \mathcal{X}^D$ as the feature vector of vertex i, where D is the number of features associated with each vertex. For the datasets we analyse, we deal with binary feature flags so $\mathcal{X} = \{0, 1\}$. We write X for the $N \times D$ *feature matrix* containing the feature vectors $\{x_i\}_{i=1}^N$ as its rows.

For the FFBM, we start with the feature matrix X and generate a random vector of block memberships $b \in [B]^N$. For each node i, the block membership $b_i \in [B]$ is generated based on the feature vector x_i, independently between nodes. The conditional distribution of b_i given x_i also depends on a collection of weight vectors $\theta = \{w_k\}_{k=1}^B$, where each w_k has dimension D. Specifically, the distribution of b given X and θ is,

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

This is the form of a softmax; we deliberately exclude a bias term to ensure that the relationships we model are based on features and not information about the size of each detected block. We will later also find it convenient to write the parameters θ as a $B \times D$ matrix of weights W.

More complex models based on different choices for the distributions ϕ_{b_i} above are also possible, but then deriving meaning from the inferred parameter distributions is more difficult.

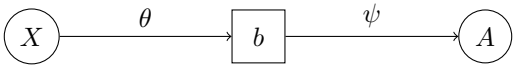


Figure 1: The feature-first block model (FFBM)

Once the block memberships b have been generated, we then draw the graph A from the microcanonical DC-SBM with additional parameters $\psi = \{\psi_e, \psi_k\}$:

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

3.1 Prior selection

To complete the description of our Bayesian framework, priors on θ and ψ must also be specified. We place a Gaussian prior on θ with zero mean and covariance matrix $\sigma_\theta^2 I$, so that each element of θ has an independent $\mathcal{N}(0, \sigma_\theta^2)$ prior, for a hyperparameter σ_θ^2:

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

Interestingly, this choice of prior leads to a particularly simple distribution on the block membership vector b. We show in Appendix B.1 that, after integrating out θ, b is uniformly distributed:

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

This is an important simplification as evaluating $p(b|X)$ does not require an expensive Monte Carlo integration over θ nor does it require the exact value of X. Peixoto [13] proposes careful choices for the additional microcanonical SBM parameters ψ, which we adopt. The idea is to write the joint distribution on (b, e, k) as a product of conditionals, $p(b, e, k) = p(b)p(e|b)p(k|e, b) = p(b)p(\psi|b)$. In our case, conditioning on X is also necessary, leading to,

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

where we used the fact ψ and X are conditionally independent given b. The exact form of the prior $p(\psi|b)$ is described in Appendix A.1. All that concerns the main argument is that it has an easily computable form.

4 Inference

Having completed the definition of the FFBM, we wish to leverage it to perform inference. Given a vertex-labelled graph (A, X), the first step is to sample from the posterior distribution of θ:

$$\theta \sim p(\theta|A, X). \tag{8}$$

We propose an iterative approach to obtain samples $\theta^{(t)}$ from $p(\theta|A, X)$. We first draw samples $b^{(t)}$ from the block membership posterior distribution, and then use each $b^{(t)}$ to obtain a corresponding sample $\theta^{(t)}$:

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

$$\theta^{(t)} \sim p(\theta|X, b^{(t)}). \tag{10}$$

Both of these sampling steps can be implemented using Markov chain Monte Carlo via a Metropolis-Hastings algorithm [5], as a two-level Markov chain.

As we show in the following section, the resulting samples for $\theta^{(t)}$ are unbiased in the sense that the expectation of their distribution is the target posterior distribution:

$$\mathbb{E}_{b^{(t)}}\left[p\left(\theta|X, b^{(t)}\right)\right] = \sum_{b\in[B]^N} p(\theta|X, b)p(b|A, X) = \sum_{b\in[B]^N} p(\theta, b|A, X) = p(\theta|A, X). \tag{11}$$

This is an example of a pseudo-marginal approach. Indeed, Andrieu and Roberts [3] show that (11) is sufficient to prove that, for large enough t, $\theta^{(t)} \sim \mathbb{E}_{b^{(i)}}\left[p(\theta|X, b^{(t)})\right] = p(\theta|A, X)$, as required.

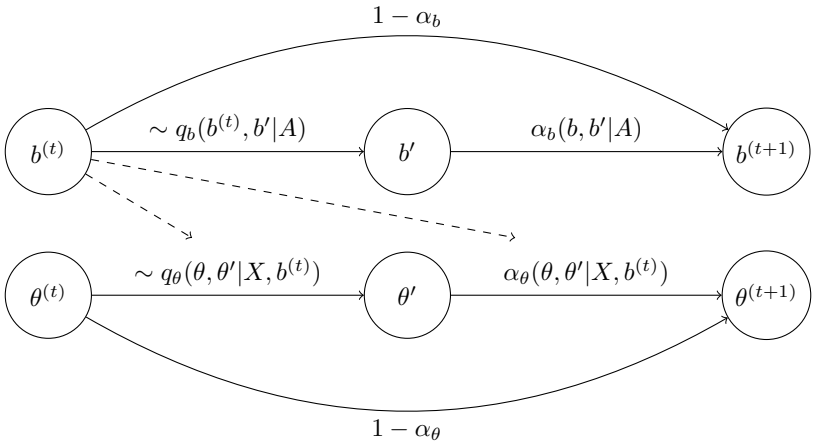


Figure 2: Sampling sequence.

The reason for splitting the Markov chain into two levels is because the summation over all latent states $b \in [B]^N$ required to directly compute the likelihood, $p(A|X, \theta) = \sum_{b\in[B]^N} p(A|b)P(b|X, \theta)$, is intractable, as the sum involves B^N terms. Figure 2 shows an overview of the proposed method. The Metropolis-Hastings accept/reject probabilities for the proposed b and θ samples are denoted α_b and α_θ, respectively.

Note the importance of the simplification in (6). As $p(b|X)$ in fact does not depend on X, knowing X is not necessary in order to obtain samples of the block membership vector b. And on the other level, in order to obtain samples of the parameter vector θ we use only b but not A, as θ and A are conditionally independnet given b.

4.1 Sampling block memberships

We adopt the Markov chain Monte Carlo procedure of [11], which relies on writing the posterior in the following form:

$$p(b|A, X) \propto p(A|b, X) \cdot p(b|X) = \pi_b(b). \tag{12}$$

Now $\pi_b(\cdot)$ is the un-normalised target density, which can be expressed as:

$$\pi_b(b) = p(b|X) \sum\nolimits_{\psi} p(A, \psi|b, X) = p(b|X)p(A, \psi^*|b, X) = p(A|b, \psi^*) \cdot p(\psi^*|b) \cdot p(b|X). \quad (13)$$

Since we are using the microcanonical SBM formulation, there is only one value of ψ that is compatible with the given (A, b) pair; recall the constraints in (1). We denote this value $\psi^* = \{\psi_k^*, \psi_e^*\}$. Therefore, the summation over all ψ reduces to just the single ψ^* term. We also define the microcanonical entropy of the configuration as:

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

This entropy can be thought of as the optimal "description length" of the graph. The exact form of the proposal q_b is explored thoroughly in [11] and not repeated here. There is a widely used library for Python made available under LGPL called `graph-tool` [12], which implements this algorithm. The only modification we make is in the block membership prior $p(b)$ that we replace with $p(b|X) = B^{-N}$, which cancels out in the MH accept-reject step as it is independent of b.

4.2 Sampling feature-to-block generator 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,

$$\pi_\theta(\theta) \propto p(\theta|X, b) \propto p(b|X, \theta)p(\theta) \propto \pi_\theta(\theta) \propto \exp\left(-U(\theta)\right), \quad (15)$$

where we write $U(\theta)$ for the negative log-posterior. Let $y_{ij} \coloneqq \mathbb{1}\{b_i = j\}$ and $a_{ij} \coloneqq \phi_j(x_i; \theta)$. Discarding constant terms, we can write $U(\theta)$ as,

$$U(\theta) = \left(\sum_{i=1}^{N} \sum_{j=1}^{B} y_{ij} \log \frac{1}{a_{ij}} \right) + \frac{1}{2\sigma_\theta^2} \|\theta\|^2 = N \cdot \mathcal{L}(\theta) + \frac{1}{2\sigma_\theta^2} \|\theta\|^2; \quad (16)$$

see Appendix B.2 for the derivation. The function $U(\theta)$ is a typical objective function for neural network training. The first term is introduced by the likelihood. We collect it into $N \cdot \mathcal{L}(\theta)$, which is the cross-entropy between the graph-predicted and feature-predicted block memberships summed over all vertices. The second, introduced by the prior, brings a form of regularisation, guarding against over-fitting. Unlike in many applications where the goal is to find the minimiser of $U(\theta)$, our goal is to draw samples from the posterior $\pi_\theta(\cdot) \propto \exp(-U(\cdot))$. 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) [14]. Given the current sample θ, we generate a new sample θ' as,

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

where $\xi \sim \mathcal{N}(0, I)$ and h is a step-size parameter which may vary with the sample index (see Appendix A.2). This leads to the proposal distribution,

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

Without the injected noise term, MALA is equivalent to gradient descent. We require the noise term ξ to fully explore the parameter space. The term ∇U has an easy to compute analytic form (derived in Appendix B.3). By noting that $\theta = \{w_k\}_{k=1}^{B}$, we write the derivative with respect to each w_k as:

$$\frac{\partial U}{\partial w_k} = -\left(\sum_{i=1}^{N} \Big\{ \tilde{x}_i (y_{ik} - a_{ik}) \Big\} - \frac{w_k}{\sigma_\theta^2} \right). \quad (17)$$

After a proposed move is generated from q_θ, it is either accepted or rejected in the standard Metropolis-Hastings accept/reject step.

4.3 Sampling sequence

Up to this point, 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 for 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 only on the matrix $y^{(t)}$ with entries $y_{ij}^{(t)} \coloneqq \mathbb{1}\{b_i^{(t)} = j\}$. We would rather deal with the expectation of each $y_{ij}^{(t)}$:

$$\mathbb{E}\left[y_{ij}^{(t)}\right] = \mathbb{E}_{b^{(t)}}\left[\mathbb{1}(b_i^{(t)} = j)\right] = p(b_i = j|A, X). \tag{18}$$

We can obtain an unbiased estimate for this quantity using the thinned b-samples after burn-in; the length of the burn-in and the details of the thinning are described in Appendix A.3. Let $\mathcal{T}_b$ denote the retained set of indices for the b-samples and $\mathcal{T}_\theta$ similarly for the θ-chain. The unbiased estimate for $y_{ij}^{(t)}$ using the restricted sample set $\mathcal{T}_b$ is denoted $\hat{y}_{ij}$:

$$\hat{y}_{ij} \coloneqq \frac{1}{|\mathcal{T}_b|}\sum_{t\in\mathcal{T}_b} y_{ij}^{(t)} = \frac{1}{|\mathcal{T}_b|}\sum_{t\in\mathcal{T}_b} \mathbb{1}\{b_i^{(t)} = j\}. \tag{19}$$

The same matrix $\hat{y}$ is used in the update step for each $\theta^{(t)}$. This way, it is not necessary to run the b and θ Markov chains concurrently. Instead, we run the b-chain to completion and use it to generate $\hat{y}$. This affords us the flexibility to vary the lengths of the b and θ-chains. Furthermore, the changeover from $y^{(t)}$ to $\hat{y}$ reduces the burn-in time for the θ-chain by reducing the variance in the evaluation of U and ∇U. A detailed description of the overall algorithms is given in Appendix C.1.

4.4 Dimensionality reduction

Once the samples $\left\{\theta^{(t)}\right\} \sim p(\theta|A, X)$ have been obtained, 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:

$$\hat{\mu}_{ij} \coloneqq \frac{1}{|\mathcal{T}_\theta|}\sum_{t\in\mathcal{T}_\theta} W_{ij}^{(t)} \qquad \text{and} \qquad \hat{\sigma}_{ij}^2 \coloneqq \frac{1}{|\mathcal{T}_\theta|}\sum_{t\in\mathcal{T}_\theta}\left(W_{ij}^{(t)} - \hat{\mu}_{ij}\right)^2 \tag{20}$$

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 (21) and only keep features with indices $d \in \mathcal{D}'$, where $\mathcal{D}'$ is given in (22).

$$\mathcal{F}_i(j) \coloneqq (\hat{\mu}_{ij} - k\hat{\sigma}_{ij}, \hat{\mu}_{ij} + k\hat{\sigma}_{ij}) \cap (-c, +c), \tag{21}$$

$$\mathcal{D}' \coloneqq \{j \in [D] : \exists i \in [B] \text{ s.t. } \mathcal{F}_i(j) = \emptyset\} \tag{22}$$

Intuitively, this means discarding any feature j for which $\hat{\mu}_{ij} \pm k\hat{\sigma}_{ij}$ lies outside the region $(-c, c)$ for all block indices i. If we were to use the Laplace approximation for the posterior $p(W_{ij}|A, X) \approx \mathcal{N}(W_{ij}; \mu_{ij}, \sigma_{ij}^2)$, then this is analogous to a hypothesis test on the value of W_{ij} as in (23); then $\mathcal{D}'$ comprises all features i for which H_1 is not rejected at least once for some $j \in [B]$

$$H_0 : |W_{ij}| \leq c \qquad H_1 : |W_{ij}| > c \tag{23}$$

The multiplier k in equation 21 determines the degree of significance of the result. Appendix **??** explores the derivation in more detail. However, as the Laplace 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 24. This approach is often preferred as it fixes the number of reduced dimensions.

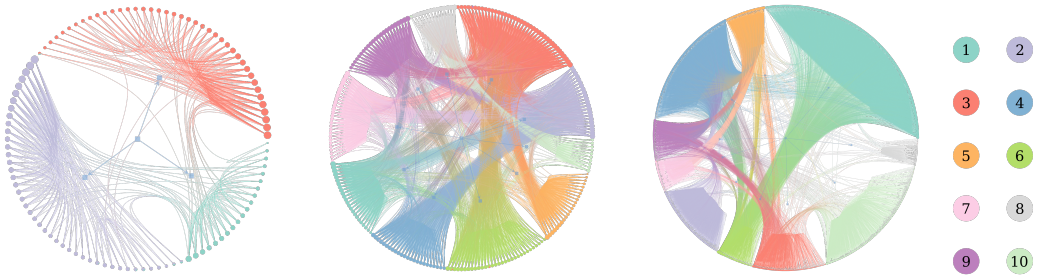
$$c^* = \arg\max\{c > 0 \ : \ |\mathcal{D}'| = D', k = k_0\}. \tag{24}$$

For an algorithmic implementation of this method refer to appendix C.1. This approach has the advantage that it fixes the number of reduced dimensions.

5 Experimental results

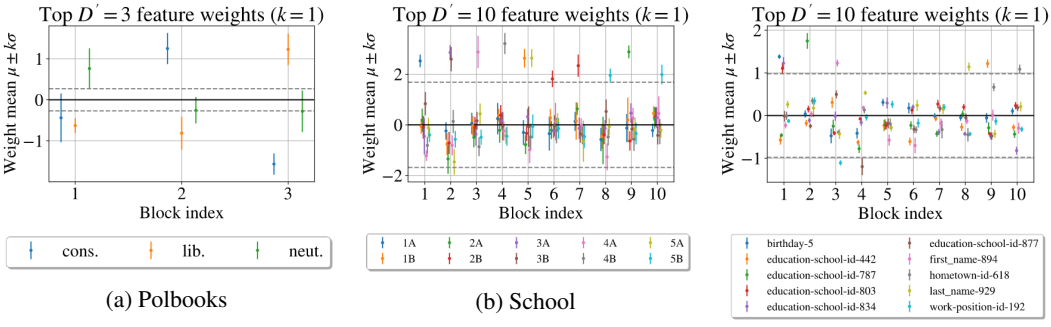
We apply the developed methods to a variety of datasets. These are chosen to span a range of node counts N, edge counts E and feature space dimension D. We consider the following:

- **Political books** [9] ($N = 105, E = 441, D = 3$) – 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 customers. Vertex features encode the political affiliation of the author (liberal, conservative, or neutral).
- **Primary school dynamic contacts** [15] ($N = 238, E = 5539, D = 13$) – 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 school-day. Vertex features include class membership (one of 10 values: 1A-5B), gender (male, female) and teacher status encoded as an 11th school-class. No further identifiable information is retained. We choose to analyse just the second day of results.
- **Facebook egonet** [6] ($N = 747, E = 30025, D = 480$) – an assortment of Facebook users' friends lists. Vertex features are extracted from each user's profile and are fully anonymised. They include information about education history, languages spoken, gender, home-town, birthday etc. We focus on the egonet with id 1912.



(a) Polbooks (b) School (c) Facebook egonet (d) Legend

Figure 3: Networks laid out and coloured according by inferred block memberships $\hat{y}$ for a given experiment iteration. Visualisation performed using *graph-tool* [12].



(a) Polbooks (b) School (c) Facebook egonet

Figure 4: Reduced dimension feature-to-block generator weight samples.

We require metrics to assess performance. 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).

Starting with the SBM, recall that the quantity $S(b)$ in (14) can be interpreted as an ideal "description length" of the partition imposed by b. We define a simple metric $\bar{S}_e$ to gauge the fit of the SBM,

Table 1: Experimental results averaged over $n = 10$ iterations (mean $\pm$ standard deviation).

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 ± 0.000	0.563 ± 0.042	0.595 ± 0.089	–	–	–
School	10	13	10	1.894 ± 0.004	0.787 ± 0.127	0.885 ± 0.129	1.198 ± 0.249	0.793 ± 0.132	0.853 ± 0.132
FB egonet	10	480	10	1.626 ± 0.003	1.326 ± 0.043	1.538 ± 0.069	0.94 ± 0.019	1.580 ± 0.150	1.605 ± 0.106

as the description length per entity (i.e., divided by the total number $N + E$ of nodes plus edges), averaged over the b-samples:

$$\bar{S}_e \coloneqq \frac{1}{(N+E)|\mathcal{T}_b|} \sum_{t \in \mathcal{T}_b} S\left(b^{(t)}\right). \tag{25}$$

Next, in order to assess the performance of the feature-to-block predictor, we partition the vertex set $[N]$ into training and test sets. We choose to randomly partition the vertices on each experiment run, so that a constant fraction f of the available vertices go to form the training set $\mathcal{G}_0$ and the remainder are held out to form the test set $\mathcal{G}_1$. The b-chain is run using the whole network but we only use vertices $v \in \mathcal{G}_0$ to train the θ-chain. Because $|\mathcal{G}_0| \neq |\mathcal{G}_1|$ in general, we cannot use the un-normalised log-target U from (16) for comparison, as the total cross-entropy loss scales with the size of each data set but the prior term stays constant. We therefore use the average cross-entropy loss over each set,

$$\bar{\mathcal{L}}_\star \coloneqq \frac{1}{|\mathcal{T}_\theta|} \sum_{t \in \mathcal{T}_\theta} \mathcal{L}_\star\left(\theta^{(t)}\right), \quad \text{where} \quad \mathcal{L}_\star\left(\theta^{(t)}\right) \coloneqq \frac{1}{|\mathcal{G}_\star|} \sum_{i \in \mathcal{G}_\star} \sum_{j \in [B]} \hat{y}_{ij} \log \frac{1}{\phi_j\left(x_i; \theta^{(t)}\right)}, \tag{26}$$

and where $\star \in \{0, 1\}$ indicates whehter the training or test set is being considered.

Table 1 summarises the results for each experiment; a complete list of all the hyperparameters used is given in Appendix C.2. We also apply the dimensionality reduction method of Section 4.4 to the two higher dimensional datasets (the school and FB egonet). For this we use equation (24) with $k = 1$, in order to reduce the dimension from D to a desired D' with $k = 1$ based on the maximal cutoff c^*. We then retrain the feature-block predictor using only the retained feature set $\mathcal{D}'$, and report the log-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 shown in Table 1.

Based on these results, some remarks are in order. Firstly, the variance of the test loss $\bar{\mathcal{L}}_1$ tends to be higher than the training loss $\bar{\mathcal{L}}_0$. This is expected, as the test set is smaller than the training set and hence more susceptible to variability in its construction. Indeed, most of the variance in the evaluation of $\bar{\mathcal{L}}_0$ and $\bar{\mathcal{L}}_1$ comes from the random partitioning of the graph into training and test sets. Secondly, it can be seen that the dimensionality reduction procedure brings the training and test losses closer together. This implies that the features we keep are indeed correlated with the underlying graphical partition and that the approach generalises correctly.

The average description length per entity, $\bar{S}_e$, of the graph, has very small variance, suggesting that the detected communities can be found reliably (to within an arbitrary relabelling of blocks). For reference, we plot an inferred partition for each of the graphs in Figure 3. The polbooks graph yields the cleanest separation between blocks but nonetheless the inferred partitions for the other datasets do succeed at partitioning the graph into densely connected clusters.

5.1 Political books

The goal here is to determine whether the authors' political affiliations are 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). From Figure 4a we see that all 3 blocks have a distinct political affiliation as their largest positive component. This strongly indicates that political affiliation is indeed the axis which best predicts the 3-way natural partition of the graph into blocks. Furthermore, in Table 1 we see that the training and test losses are very similar and both are low in magnitude. This provides further evidence to the claim that political affiliation is a very appropriate explanatory variable for the overall network structure.

5.2 Primary school dynamic contacts

We choose the number of communities $B = 10$, in line with the total number of school classes. As before, we sample the block-generator parameters θ and employ the dimensionality reduction technique of Section 4.4, with standard deviation multiplier $k = 1$ to pick out the top $D' = 10$ features. We then plot the weights for the resulting features $d \in \mathcal{D}'$ in Figure 4b. It is rather obvious that only the pupils' class memberships (1A-5B) have remained significant; 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 two comparably good classes as their predictor. For example, block 2 contains classes 3A and 3B as its 2 best predictors. This suggests that the social divide between classes is less pronounced for pupils in year 3. Conversely, some classes are found to extend over two detected blocks (class 2B spans blocks 6 and 7) but we nonetheless do not have a feature which explains the division. The most 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

We choose $B = 10$ and $D' = 10$ for this experiment. The selected features (Figure 4c) are those that best explain the high-level community structure. The majority of them are education related. Nevertheless, for $D' = 10$ we only have good explanations for the makeup of some of the detected blocks; several blocks in Figure 4c do not have high-magnitude components for $D' = 10$.

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 prior goes some way towards alleviating this problem. Nevertheless, we see in Figure 4c that the feature `birthday-5` has a very high weight as it relates to block 1 – but it is highly unlikely that birthdays determine graphical structure.

6 Conclusion

The Feature-First Block Model (FFBM) introduced in this paper is a new generative model for labelled networks, developed in order to address difficulties 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 test whether the vertex features can accurately explain this partition. It is relatively 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.

Using this new model, we go on to describe an efficient inference algorithm to sample the parameters θ of the feature-to-block generator. This takes the form of two concurrent Markov chains, used to sample the block memberships b and block generator parameters θ. These chains can either be executed in parallel or the empirical mean of the b-samples can be used as the input to the θ-chain. The latter option reduces the variance in our evaluation of the target distribution and thus shortens burn-in.

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 block. 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 several scales of interest. So long as data collection techniques remain ethical and care is taken to respect personal privacy, such empowered decision-making can only help humankind.

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] 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.

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

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

[13] 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.

[14] 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/.

[15] 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`.

[16] 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.

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

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See experiments section 5
   (b) Did you describe the limitations of your work? [Yes] See conclusion 6 and experiments 5 sections
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See conclusion 6
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See inference section 4
   (b) Did you include complete proofs of all theoretical results? [Yes] See inference 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)? [Yes] See supplementary code
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See appendix C.2
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See table 1
   (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)? [Yes] See appendix C.3
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? [Yes] See section experiments 5
   (b) Did you mention the license of the assets? [Yes] See section inference 4
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Supplementary material
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] Referred to original papers
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] Referred to original papers
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]

Appendices

A Additional material

A.1 SBM prior choice explanation

Here we recall for reference the priors $p(\psi|b)$ from [13]:

$$p(\psi_e = e, \psi_k = k|b) = p(e|b)p(\psi_k|e,b) = \left[\left\{\!\!\left\{ {B \atop 2} \right\}\!\!\right\}^{-1}_{E}\right] \cdot \left[\prod_r \frac{\prod_j \eta_j^r!}{n_r! q(e_r, n_r)}\right], \tag{27}$$

where $\left\{\!\!\left\{ {n \atop m} \right\}\!\!\right\}$ 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 produced by m samples in n bins. The value $E = \frac{1}{2}\sum_{r,s} e_{rs}$ is the total number of edges in the graph. Importantly, E is not allowed to vary and so $p(e|b)$ is uniform in e. The variable η_j^r denotes the number of vertices in block r that have degree j; formally, $\eta_j^r \coloneqq \sum_i \mathbb{1}\{b_i = r\}\,\mathbb{1}\{k_i = j\}$. The denominator $q(m,n)$ denotes the number of different histograms produced by m samples in at most n non-zero bins that sum to m. Finally, $e_r \coloneqq \sum_s e_{rs}$ 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.

These were chosen carefully in [13] to more closely match the structure of empirical networks than simple uniform priors. We do not repeat these arguments here.

A.2 Choosing the MALA step-size

Recall that in Section 4.2 we used the Metropolis Adjusted Langevin Algorithm (MALA) in order to sample from the θ-chain of the block membership generator parameters. At iteration t, the proposed sample is generated by:

$$\theta' = \theta^{(t)} - h_t \nabla U(\theta^{(t)}) + \sqrt{2h_t}\cdot\xi. \tag{28}$$

There are two competing objectives when choosing the step-size h_t. On the one hand, h_t needs to be large so that the sampler arrives at a high density region quickly, while too large a step-size would lead to low acceptance rates and thus inefficient sampling. An effective strategy is to use *simulated annealing*: allow h_t to slowly decrease with t, as long as $h_t > 0$ for all t and also:

$$\sum_{t=1}^{\infty} h_t = \infty, \qquad \text{and} \qquad \sum_{t=1}^{\infty} h_t^2 < \infty. \tag{29}$$

Following Welling and Teh [16], we adopt the polynomially decaying step-sizes, $h_t = \alpha(\beta + t)^{-\gamma}$, where $\alpha > 0$, $\beta > 0$ and $\gamma \in (1/2, 1]$ are hyper-parameters. We make the specific choices,

$$\alpha = \frac{250 \cdot s}{N}, \qquad \beta = 1000, \qquad \gamma = 0.8, \tag{30}$$

where N is the number of data-points and s, the *step-size scaling*, is the only free parameter.

A.3 Burn-in and thinning

When sampling from the b- and θ-chains described in Section 4, we generate T_b and T_θ samples total, respectively. We discard an initial proportion $\kappa_\star \in (0,1)$ of the samples as corresponding to a "burn-in" period required for the distribution of the chain to reach a distribution close to our target, and we also "thin" the remaining samples to obtain a less-dependent version. For $\star \in \{b, \theta\}$, the remaining sample sets are denoted $\mathcal{T}_\star$ in the notation of Section 4.3,

$$\mathcal{T}_\star = \{T_\star\kappa_\star + i\lambda_\star : 0 \le i \le \lfloor T_\star(1-\kappa_\star)/\lambda_\star \rfloor\}, \tag{31}$$

where $\lambda_\star$ controls the thinning. The choice of $\kappa_\star$ can be determined by plotting the log-target (either $S(b^{(t)})$ or $U(\theta^{(t)})$ as a function of t, and choosing $\kappa_\star$ to encompass the region where the log-target has roughly reached equilibrium. As we do not leverage sample independence, $\lambda_\star$ can be chosen less rigorously; we often simply use $\lambda_b = 5$ and $\lambda_\theta = 10$.

A.4 Initializing the b-chain

For the purposes of the FFBM model, the number of blocks B is a constant which must be specified. If the choice of B is influenced by the observed data, then the analysis is no longer "fully Bayesian" and belongs to the class of methods referred to as "empirical Bayes." However, as the number of blocks only specifies the coarseness of the analysis, it is reasonable to allow it to vary. Indeed, Peixoto [10] 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 B is allowed to vary in the b-chain (i.e., when new blocks can be created and empty blocks are allowed), then the chain can be run until a minimum description length (MDL) solution is reached. We take the number of non-empty blocks in the MDL solution to be our fixed block number B for subsequent analysis. Indeed, it is prudent to start the b-chain at this MDL solution as then the necessary burn-in time can be greatly reduced.

B Derivations

B.1 Derivation of conditional block distribution given feature matrix

We determine the form of $p(b|X)$ by integrating out the parameters θ. From the definitions we have:

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

The key observation here is that the value of the integral is independent of the value of $b_i \in [B]$: The integrand has the same form regardless of b_i, because the prior is the same for each w_j. Therefore, the integral is only a function of $\tilde{x}_i$ and σ_θ^2, which means that, as function of b, $p(b|X) \propto 1$. And as b takes values in $[B]^N$, we necessarily have:

$$p(b|X) = \frac{1}{\left|[B]^N\right|} = B^{-N}. \tag{32}$$

B.2 Derivation of $U(\theta)$

Recall from (15) in Section 4.2 that,

$$\pi_\theta(\theta) \propto p(\theta|X,b) \propto p(b|X,\theta)p(\theta) \propto \exp\left(-U(\theta)\right),$$

so that U can be expressed as,

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

Writing, $y_{ij} \coloneqq \mathbb{1}\{b_i = j\}$ and $a_{ij} \coloneqq \phi_j(x_i;\theta)$, we have,

$$\log p(b|X,\theta) = \sum_{i\in[N]}\sum_{j\in[B]} y_{ij}\log a_{ij} \quad \text{and} \quad \log p(\theta) = -\frac{(D+1)(B)}{2}\log 2\pi - \frac{1}{2\sigma_\theta^2}\|\theta\|^2, \tag{33}$$

where $\|\theta\|^2 = \sum_i \theta_i^2 = \sum_{j=1}^B \|w_j\|^2$ is the Euclidean norm of the vector of parameters θ. Therefore, discarding constant terms, we obtain exactly the representation (16), as claimed.

B.3 Derivation of $\nabla U(\theta)$

Here we show how the gradient $\nabla U(\theta)$ can be computed explicitly. Recall the expression for $U(\theta)$ in (16). Writing θ as $\theta = \left[w_1^T, w_2^T \dots w_B^T\right]^T$, in order to compute the gradient $\nabla U(\theta)$ we need to compute each of its components, $\partial U/\partial w_k$, $1 \le k \le B$. To that end, we first compute,

$$\begin{aligned} \frac{\partial a_{ij}}{\partial w_k} &= \frac{\tilde{x}_i\exp(w_j^T\tilde{x}_i)\delta_{jk}\cdot\sum_{r\in[B]}\exp(w_r^T\tilde{x}_i) - \exp(w_j^T\tilde{x}_i)\cdot\tilde{x}_i\exp(w_k^T\tilde{x}_i)}{\left(\sum_{r\in[B]}\exp(w_r^T\tilde{x}_i)\right)^2} \\ &= \tilde{x}_i\left(a_{ij}\delta_{jk} - a_{ij}a_{ik}\right), \end{aligned} \tag{34}$$

where $\delta_{jk} \coloneqq \mathbb{1}\{j = k\}$, and we also easily find,

$$\frac{\partial}{\partial w_k}\|\theta\|^2 = \frac{\partial}{\partial w_k}\left(\sum_{r\in[B]}\|w_r\|^2\right) = 2w_k. \tag{35}$$

Using (34) and (35), we obtain,

$$\begin{aligned}\frac{\partial U}{\partial w_k} &= \sum_{i=1}^{N}\sum_{j=1}^{B} y_{ij}\left(-\frac{\tilde{x}_i}{a_{ij}}\left(a_{ij}\delta_{jk} - a_{ij}a_{ik}\right)\right) + \frac{w_k}{\sigma_\theta^2} \\ &= -\left(\sum_{i=1}^{N}\tilde{x}_i\left(y_{ik} - a_{ik}\sum_{j=1}^{B} y_{ij}\right) - \frac{w_k}{\sigma_\theta^2}\right) \\ &= -\left(\sum_{i=1}^{N}\left\{\tilde{x}_i(y_{ik} - a_{ik})\right\} - \frac{w_k}{\sigma_\theta^2}\right). \end{aligned} \tag{36}$$

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

C Computational details

C.1 Algorithms

Algorithm 1 Block membership sample generation

$b^{(0)} \leftarrow \arg\min_b S(b|A)$ ▷ Implemented as greedy heuristic in *graph-tool* library
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

$\hat{Y}_{ij} \leftarrow \frac{1}{|T_b|} \sum_{t \in T_b} \mathbb{1}\{b_i^{(t)} = j\} \quad \forall i, j$
$\theta^{(0)} \leftarrow\sim \mathcal{N}(0, \sigma_\theta I)$

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 $\{\theta^{(t)}\}_{t=1}^{T_\theta}$

C.2 Hyperparameter values

Table 2: Hyper-parameter 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	0.7	1	1,000	0.2	5	10,000	0.4	10	0.05	–	–	–	–	–	–
School	10	0.7	1	1,000	0.2	5	10,000	0.4	10	0.2	1	10	10,000	0.4	10	0.2
FB Egonet	10	0.7	1	1,000	0.2	5	10,000	0.4	10	0.017	1	10	10,000	0.4	10	0.5

C.3 Implementation details

All data analysis and visualisation was implemented in Python. Full source code is available in the supplementary material. The scripts were run using a standard PC using the Windows Subsystem for Linux (WSL) environment. Specs are:

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

On this hardware each experiment iteration took the following amount of time to execute:

Table 3: Compute-time for each experiment.

Dataset	b-chain	θ-chain	Reduced θ-chain	Overall compute time
Polbooks	$\sim$1s	$\sim$4s	–	$\sim$5s
School	$\sim$10s	$\sim$10s	$\sim$10s	$\sim$30s
FB Egonet	$\sim$20s	$\sim$180s	$\sim$10s	$\sim$210s