# Inferring community characteristics in labelled networks

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

**Ioannis Kontoyiannis**
Department of Mathematics
University of Cambridge
ik355@cam.ac.uk

## Abstract

We present a novel generative model for describing vertex-labelled undirected graphs. This is a two-layered model. First a probabilistic mapping from vertex labels to latent block memberships and from these block memberships we generate a graph according to the micro-canonical stochastic block model.

With this framework we are able to efficiently sample the parameters of the vertex label to block membership mapping. This allows us to automatically determine which features have the largest impact on graphical structure.

## 1 Introduction

There is a wealth of networks in the world. Some examples of this graphical data are social networks, website hyperlinks and academic collaboration with more being produced each second. It is clear we need tools to analyse this increasingly omni-present form of data.

A somewhat surprising property of real-world graphs is that they exhibit strong community structure. In other words, each node will often belong to a cluster of densely connected nodes. This property is often exploited by graph compression algorithms and there is high interest in recovering the communities from the observed graph.

A common subset of graphical data is the labelled network. This is a graph where we have information about the properties of each node. We shall refer to these node properties as features. One of the most common questions we can ask of this dataset is what features have the largest impact on the structure of the graph. For example, when analysing an academic collaboration graph, one may wish to ask what impact gender has on the structure of the network. Nevertheless, this analysis is often vulnerable to confounding variables. While gender very well may impact the structure of the graph, there is often a better explanatory variable for the structure.

There is space to bridge the gap between these two approaches. Rather than determining whether a feature impacts the graphical structure directly, we can use the community concept as a stepping stone in our analysis. Therefore, we extract which features have the largest impact on overall graphical structure. This can be thought of as a form of dimensionality-reduction where we only keep the node features that are useful in predicting community memberships.

## 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 this ensemble exhibit community

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

structure. Specifically, we will use the microcanonical variant of the SBM, proposed by Peixoto [8]. A paraphrased definition is given below.

**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) 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 Latent block generative 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 latent block generative model - LBGM) 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 $\theta$. 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 $\theta$ 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 $\theta$ 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.

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 $\psi$.

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

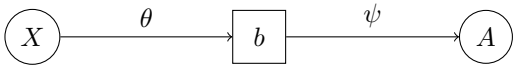


Figure 1: Latent block generative model

### 3.1 Prior selection

Before performing any inference, we must specify priors on $\theta$ and $\psi$. For $\theta$ it seems sensible to choose a Gaussian prior, with zero mean and variance matrix $\sigma_\theta^2 I$ such that each element of $\theta$ is independent and distributed like $\sim \mathcal{N}(0, \sigma_\theta^2)$. In vector form, the prior for $\theta$ 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 [8] proposes careful choices for the additional microcanonical SBM parameters $\psi$ 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 $\psi$, 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 [8] for $p(\psi|b)$, repeated here for reference.

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

Where $\left\{\!\!\left\{ {n \atop m} \right\}\!\!\right\}$ 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 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 $\eta_j^r$ 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 $\left\{\!\!\left\{ {n \atop m} \right\}\!\!\right\}$. 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 LBGM, 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 $\theta$ 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 $\theta$ 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 $\theta$ 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 [3]. 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 $\alpha$ (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 [1]. 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

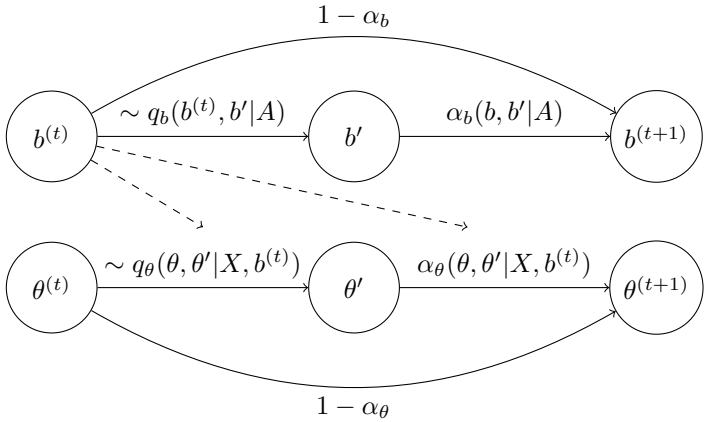


Figure 2: Sampling sequence

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 [6] 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 $\pi_b$ 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 $\psi$ 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 $\psi$ reduces to just the single $\psi^*$ 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 [6]. There is a widely available library, `graph-tool` [7], implementing 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 $\theta$-chain.

### 4.2 Sampling feature-to-block classifier parameters

The invariant distribution we wish to target for the $\theta$ samples is the posterior of $\theta$ 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 **??**). 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 $\theta$.

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

$U(\theta)$ in equation 18 appears a typical objective function to be minimised for neural network training. The first term is the cross-entropy between actual and predicted labels. The second term - introduced by the prior - brings a form of regularisation, preventing over-fitting. In traditional applications we only seek the value of $\theta$ 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 $\nabla U$ 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 $\nabla U$ 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 [9] - which leverages just that. Given the current sample $\theta$, we propose a new sample $\theta'$ 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 $\nabla U$ 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 $\alpha_\theta$.

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

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 $\theta$ 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 $\theta$ Markov chain. This affords us the flexibility to vary the number of samples we draw for $b$ and $\theta$; wed refer to these as $T_b$ and $T_\theta$ henceforth. Furthermore, this changeover reduces the burn-in time for the $\theta$-chain by reducing the variance in our evaluation of $U$ and $\nabla U$.

### 4.4 Complexity Analysis

## 5 Experiments

We apply the developed methods to a variety of datasets.

**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** LBGM parameter pseudo-marginal inference

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

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

- **College football teams** [2] - network of American college football teams and their interactions. Vertex labels are the division a team belongs to.
- **Law firm** [4] - a network of relationships between members of a law firm. Each relationship is categorised according to type: coworkers, friends or advice.
- **Maier Facebook Egonet** [5] - network of the author's Facebook friends list. Each vertex has been manually labelled with a variety of features describing their relationship to the author.
- **Twitch users** [10] - a network of user-user friendships on the streaming service Twitch. Vertex labels are extracted according to video-games played, location and streaming habits. This dataset is also broken down into disjoint networks according to language. We only consider the English users with is a subnet with $N = 7126$ vertices and $E = 35324$ edges).

# 6 Extensions

# 7 Conclusion


## Acknowledgments and Disclosure of Funding

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] 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`.

[2] T S Evans. Clique graphs and overlapping communities. *Journal of Statistical Mechanics: Theory and Experiment*, 2010(12):P12037, Dec 2010. ISSN 1742-5468. doi: 10.1088/1742-5468/2010/12/p12037. URL `http://dx.doi.org/10.1088/1742-5468/2010/12/P12037`.

[3] 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`.

[4] 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.

[5] 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`.

[6] 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`.

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

[8] 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`.

[9] 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/`.

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

[11] 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.


## 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? **[TODO]**
   (b) Did you describe the limitations of your work? **[TODO]**
   (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? **[TODO]**
   (b) Did you include complete proofs of all theoretical results? **[TODO]**
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? **[TODO]**
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? **[TODO]**
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? **[TODO]**

# 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 $\theta$.

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

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 $\sigma_\theta^2$ 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:

$$p(b|X) = B^{-N} \tag{25}$$

### 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 26.

$$U(\theta) = \left( \sum_{i=1}^{N} \sum_{j=1}^{B} y_{ij} \log \frac{1}{a_{ij}} \right) + \frac{1}{2\sigma_\theta^2} ||\theta||^2 \tag{26}$$

Where $y_{ij}$ is independent of $\theta$ and $a_{ij}$ is the output from the softmax layer, with form as given in equation 27.

$$a_{ij} \coloneqq \phi_j(x_i; \theta) = \frac{\exp(w_j^T \tilde{x}_i)}{\sum_{b=1}^{B} \exp(w_b^T \tilde{x}_i)} \tag{27}$$

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 $a_{ij}$:

$$\begin{aligned} \frac{\partial a_{ij}}{\partial w_k} &= \frac{\tilde{x}_i \exp(w_j^T \tilde{x}_i)\delta_{jk} \cdot \sum_{b=1}^{B} \exp(w_b^T \tilde{x}_i) - \exp(w_j^T \tilde{x}_i) \cdot \tilde{x}_i \exp(w_k^T \tilde{x}_i)}{\left(\sum_{b=1}^{B} \exp(w_b^T \tilde{x}_i)\right)^2} \\ &= \tilde{x}_i \left(a_{ij}\delta_{jk} - a_{ij}a_{ik}\right) \end{aligned} \tag{28}$$

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

$$\frac{\partial}{\partial w_k} ||\theta||^2 = \frac{\partial}{\partial w_k} \left( \sum_{b=1}^{B} ||w_b||^2 \right) = 2w_k \tag{29}$$

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

$$\begin{aligned} \frac{\partial U}{\partial w_k} &= \sum_{i=1}^{N} \sum_{j=1}^{B} y_{ij} \left( \frac{-\tilde{x}_i}{a_{ij}} \left(a_{ij}\delta_{jk} - a_{ij}a_{ik}\right) \right) + \frac{w_k}{\sigma_\theta^2} \\ &= -\left( \sum_{i=1}^{N} \tilde{x}_i \left( y_{ik} - a_{ik} \sum_{j=1}^{B} y_{ij} \right) - \frac{w_k}{\sigma_\theta^2} \right) \\ &= -\left( \sum_{i=1}^{N} \left\{ \tilde{x}_i (y_{ik} - a_{ik}) \right\} - \frac{w_k}{\sigma_\theta^2} \right) \end{aligned} \tag{30}$$

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

$$\theta' = \theta^{(t)} - h_t \nabla U(\theta^{(t)}) + \sqrt{2h_t} \cdot \xi \tag{31}$$

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 [11]:

$$\sum_{t=1}^{\infty} h_t = \infty \qquad \text{and} \qquad \sum_{t=1}^{\infty} h_t^2 < \infty \tag{32}$$

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 [11] propose the following form for a polynomially decaying step-size which we adopt:

$$h_t = a(b + t)^{-\gamma} \tag{33}$$

Where $a, b, \gamma$ are hyper-parameters to be chosen. We require $a, b > 0$ and $\gamma \in (0.5, 1]$ to satisfy equation 32. We find empirically that $a = 250, b = 1000, \gamma = 0.8$ yield good burn-in times for our MALA implementations.