Automatic feature classification in stochastic block models

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 graphical data in the world with more produced every second; social networks, website hyperlinks and academic collaboration are just some examples. Many algorithms are used to analyse this data. Nevertheless, many of the approaches have been developed in industry to yield empirically good results. There is space to develop a principled framework for Bayesian inference on graphical data.

2 Preliminaries

We will be using the microcanonical stochastic block model, proposed by [7]. A paraphrased definition is given below.

Definition 2.1 (Microcanonical SBM) *Let $N \in \mathbb{Z}^+$ denote the number of vertices in our graph. The block memberships are denoted by a vector b of length N where each entry $b_i \in \{1, 2 \ldots B\}$ where B is the number of nonempty 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 $r = s$). We restrict our analysis to undirected graphs so 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 at random but respecting the constraints imposed by e and b. The additional parameters N and B are omitted as they are inferred from the shape ofs of b and e. If we interpret A as an adjacency matrix, then this constraint can be written formally as: $e_{rs} = \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij} \mathbb{1}\{b_i = r\}\mathbb{1}\{b_j = s\}$. However, this formulation does not tolerate high degree variation within blocks as is typical of real-world data. We therefore introduce the degree-corrected SBM (DC-SBM) which has an additional parameter k which is a vector of length N encoding the degree sequence (k_i is the degree of vertex i). Therefore, we write:

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

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

This imposes the additional constraint that $k_i = \sum_{j=1}^{N} A_{ij}$. In what follows, we will always assume the degree-corrected model unless otherwise specified.

3 Latent block generative model

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 is given in figure 1. We start, with the feature matrix X and generate a vector of block memberships b. The parameters of this generator are encapsulated by θ. Each feature vector x_i is treated independently and used to generate the block membership b_i. 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 complicated. Summarising, we can 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 $\tilde{x} := [x_1, x_2, \ldots x_D, 1]^T$ is an augmented version of x that allows for a bias term. The parameters θ just contain the $B \times (D+1)$ matrix of weight values or alternatively, $\theta = \{w_k\}_{k=1}^B$.

$$\begin{tikzcd} X \arrow[r, "\theta"] & \boxed{b} \arrow[r, "\psi"] & A \end{tikzcd}$$

Figure 1: Latent block generative model

Once we have generated the block memberships b, we proceed to draw the graph A from the microcanical SBM (2.1) 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 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 $\sim \mathcal{N}(0, \sigma)$. 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. As $\sigma_\theta^2 \to \infty$ this becomes an uninformative uniform prior.

In our model b is now an intermediate variable and so we cannot choose a prior. The closest thing we can get to a prior is $p(b|X)$. 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 that our choice of prior for $p(\theta)$ leads to the following form for $p(b|X)$.

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

In his paper, Peixoto [7] proposes careful choices for the SBM parameters. The proposal 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.

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

Where it is apparent by regarding figure 1 as a Markov model that $(\psi \perp\!\!\!\perp X)|b$. We then borrow the priors proposed by Peixoto [7] for $p(\psi|b)$, repeated here for reference.

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

Where $\left\{{n \atop m}\right\}$ 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 histograms (non-negative bin values) with n bins that are constrained to sum to m. $E = \frac{1}{2}\sum_{r,s} e_{rs}$ is the total number of edges. 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, $\eta_j^r := \sum_i \mathbb{1}\{b_i = r\}\,\mathbb{1}\{k_i = j\}$. Furthermore, $q(m,n)$ is the number of different histograms with at most n non-zero bins that sum to m. Lastly, $e_r := \sum_s e_{rs}$ is the total number of half edges in block r and $n_r := \sum_i \mathbb{1} b_i = r$ 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

We are presented with a vertex-labelled graph (A, X). The goal is to draw samples from equation 9. However, this is not easily done in practice.

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

We instead propose an iterative approach. First drawing samples $b^{(i)}$ 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 can be implemented with a Markov Chain through the Metropolis-Hastings algorithm [2]. 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.

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

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 computationally intensive. Figure 2 shows an overview of the proposed method. We have introduced subscripts and conditionings to make explicit what parameters each step utilises. In an important simplification, $p(b|X) = B^{-N}$ which is unchanged no matter the value of X. Therefore, we do not need to know the value of X to perform the sampling on b.

4.1 Sampling block memberships

Peixoto [5] proposes a Monte Carlo method which we will base our approach on. It relies on writing the posterior in the following form.

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

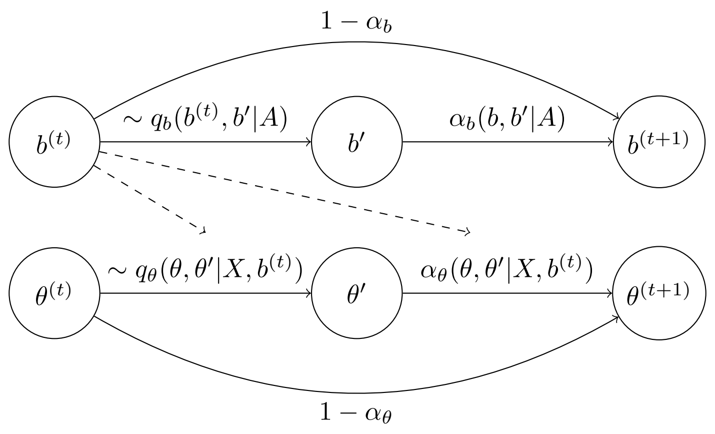
$1-\alpha_b$
$\sim q_b(b^{(t)}, b'|A)$
$\alpha_b(b, b'|A)$
$b^{(t)}$
b'
$b^{(t+1)}$
$\sim q_\theta(\theta, \theta'|X, b^{(t)})$
$\alpha_\theta(\theta, \theta'|X, b^{(t)})$
$\theta^{(t)}$
θ'
$\theta^{(t+1)}$
$1-\alpha_\theta$

Figure 2: Sampling sequence

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 a microcanonical formulation, there is only one value of ψ which we denote ψ^* that is compatible with the given (A,b) pair. Specifically, $k_i^* = \sum_j A_{ij}$ and $e_{rs}^* = \sum_{i,j} A_i j \mathbb{1}\{b_i = r\}\mathbb{1}\{b_j = s\}$. 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 [5]. There is a widely available library `graph-tool` [6] implementing this algorithm, which we use. The only modification is in the block membership prior $p(b)$ which we replace with $p(b|X) = B^{-N}$ which is a uniform distribution and so does not impact the MH accept-reject step.

4.2 Sampling feature classifier parameters

Now 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:

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

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

Where we have introduced $U(\theta)$ equal to the negative log posterior, because it simplifies analysis. Each of the constituent terms of$U(\theta)$ is easily computed. To simplify notation, we define $y_{ij} := \mathbb{1}\left\{b_i = j\right\}$ and $a_{ij} = \phi_j(x_i;\theta)$.

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

This means that $U(\theta) = H_c(t,y) + {}^1\!/_{2\sigma_\theta^2}||\theta||^2$ plus a constant, where $H_c(\cdot||\star)$ is the cross entropy between two distributions and the prior introduces a regularisation term. This is a typical objective function used in neural network training. In typical applications we only seek the value of θ that minimises the objective function, which in our case would yield the maximum a posteriori (MAP) estimate. This is often done through some kind of gradient descent. However, our goal is not to find the MAP estimate but to draw samples from the posterior. We therefore adopt the Metropolis Adjusted Langevin Algorithm (first proposed by Roberts and Tweedie [8]). Given the current sample θ, we propose a new sample θ' according to equation 18

$$\theta' = \theta - h\nabla U(\theta) + \sqrt{2h} \cdot \xi \tag{18}$$

Where $\xi \sim \mathcal{N}(0, I)$. 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 evaluated easily.

$$q_\theta(\theta, \theta') = \mathcal{N}\left(\theta'; \theta - h\nabla U(\theta), 2hI\right) \tag{19}$$

The term ∇U has an easy to compute analytic form. By noting that $\theta = \{w_k\}_{k=1}^{B}$

$$\frac{\partial U}{\partial w_k} = -\left(\sum_{i=1}^{N}\left\{\tilde{x}_i(y_{ik} - a_{ik})\right\} - w_k\right) \tag{20}$$

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

$$\alpha(\theta, \theta') = \min\left(\exp\left(-\Delta U\right)\frac{q(\theta', \theta)}{q(\theta, \theta')}, 1\right) \tag{21}$$

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. $U(\theta)$ is fed in the information $y_{ij}^{(t)} = \mathbb{1}\{b_i^t = j\}$. We would rather deal with the expectation of this value.

$$\mathbb{E}\left[y_{ij}^{(t)}\right] = \mathbb{E}\left[\mathbb{1}(b_i^{(t)} = j)\right] = p(b_i = j) \tag{22}$$

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

$$\hat{y}_{ij} \coloneqq \frac{1}{T}\sum_{t=1}^{T} y_{ij}^{T} \tag{23}$$

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 as well as reducing the burn-in time for the θ-chain.

5 Experiments

We apply the developed methods to a variety of datasets.

- **Maier Facebook Egonet** [4] - This is a 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.
- **College football teams** [1] - network of American college football teams and their interactions. Vertex labels are the division a team belongs to.
- **Law firm** [3] - a network of relationships between members of a law firm. Each relationship is categorised according to type: coworkers, friends or advice.

6 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] 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`.

[2] 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`.

[3] 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.

[4] 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`.

[5] 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`.

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

[7] 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`.

[8] 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/`.

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 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:

$$p(b|X) = B^{-N} \tag{24}$$

A.2 Derivation of gradient with respect to feature parameters