Variational Graph Convolutional Networks

Louis Tiao[*] The University of Sydney
Pantelis Elinas CSIRO's Data61
Harrison Nguyen[*] The University of Sydney
Edwin V. Bonilla CSIRO's Data61

Abstract

We propose a Bayesian approach to graph convolutional networks (GCNs) where the graph parameters are considered as random variables. We develop an inference algorithm to estimate the posterior over these parameters and use it to incorporate prior information that is not naturally considered by standard GCNs. The key to our approach is to define a smooth posterior parameterization over the adjacency matrix characterizing the graph, which we estimate via stochastic variational inference. Our experiments show that we can outperform standard GCN methods in the task of semi-supervised classification in noisy-graph regimes.

1 Introduction

In this work we address the problem of semi-supervised classification based on graph convolutional networks (GCNs), as proposed by [15]. These networks can be seen as a first-order approximation of the spectral graph convolutional networks developed by [3], which itself built upon the pioneering work of [2, 9]. The great popularity of GCNs is mainly due to their practical performance as, at the time it was published, it outperformed related methods by a significant margin. Another practical advantage of using GCNs is their relatively simple propagation rule, which does not require expensive operations such as eigen-decomposition.

However, one of the main assumptions underlying GCNs is that the given graph is helpful for the task at hand and that the corresponding edges are highly reliable. This is generally not true in practical applications, as the given graph maybe (i) noisy, (ii) loosely related to the classification problem of interest or (iii) built in an ad hoc basis using e.g. side information. In practice, it is difficult to incorporate this type of uncertainty over the graph using the original GCN framework in a principled way and the performance of the method degrades significantly with increasingly noisy graphs. In this paper we address exactly this problem by adopting a Bayesian approach that places a prior over the graph structure, as given by the adjacency matrix. This, however, poses significant inference challenges as estimating the posterior over the adjacency matrix under a highly no-linear likelihood (as given by the GCN's output) is intractable.

Thus, to estimate this posterior we resort to approximations as given by stochastic variational inference (VI), which is underpinned by the maximization of the evidence lower bound (ELBO). Nevertheless, a crucial remaining challenge is how to parameterize the posterior over $\mathcal{O}(N^2)$ parameters, where N is the number of nodes in the graph. We propose a simple but effective parameterization inspired by low-rank matrix factorization, where we assume underlying continuous latent variables for each row/column of the posterior parameters over the adjacency matrix. To the best of our knowledge, we are the first to develop an efficient variational Bayesian approach that incorporates uncertainty information about the given graph directly. Our experiments show that we can outperform standard GCN methods and other competitive baselines in the task of semi-supervised classification in noisy-graph regimes.

[*]Work done while at CSIRO's Data61.

Graph Representation Learning Workshop (NeurIPS 2019), Vancouver, Canada.

1.1 Related work

Most graph-convolutional neural network (CNN) frameworks can be seen from a more general perspective under the unifying mixture models network (MONET) [19]. Therefore, for more details of graph neural network approaches, we refer the reader to [19] as well as to the excellent related work in [26] and, more generally, to the surveys in [27, 29].

In terms of probabilistic approaches to graph neural networks, [28] proposes a probabilistic model for GCNs where the graph is considered as a realization of mixed membership stochastic block models [1]. However, despite their method being referred to as Bayesian, it parameterizes the *true posterior* over the graph directly and this posterior is not dependent on the observed data (neither features or labels). Thus, it can be seen more like an ensemble GCN. Similarly targeting the low-labeled-data regime, [20] proposes a method for semi-supervised classification with Gaussian process (GP) priors, where the parameters of a robust-max likelihood for a node are given by the average of the GP values over its 1-hop neighborhood. Contemporary to our work, [5] develops a method for learning the graph structure using a generative model and estimates its parameters via bilevel optimization. However, unlike our approach, their method is not Bayesian and it does not allow for either the incorporation of prior knowledge or estimation of the full posterior over the graph. Finally, [14] proposes the variational graph autoencoder, a probabilistic framework that also learns latent representations for graphs but, unlike our work, is designed for the task of link prediction.

2 Bayesian graph convolutional networks

Let $\mathbf{X} \in \mathbb{R}^{N \times D}$ be a set of D-dimensional features representing N instances and $\mathbf{Y} = \{\mathbf{y}_n\}$ be their corresponding labels, some of which are observed and others unobserved and $\mathbf{y}_n \in \{0,1\}^C$ is one-hot-encoded. The goal of semi-supervised classification is to leverage the labeled and unlabeled data in order to predict the unobserved labels. In this paper we are interested in doing so by explicitly exploiting the dependencies among data-points as given by a graph $\mathcal{G}$. To do this, we consider the popular GCN models as proposed by [15], which can be seen as first-order approximations to more general (but computationally costly) spectral graph convolutional networks [3, 8].

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote an undirected graph with N nodes $v_i \in \mathcal{V}$, edges $(v_i, v_j) \in \mathcal{E}$ and binary adjacency matrix $\mathbf{A} \in \{0,1\}^{N \times N}$. [15] showed that when considering various simplifying assumptions, one can rewrite, for a signal $\mathbf{X}$, the convolved signal matrix as $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}\mathbf{W}$, where $\mathbf{W}$ is a matrix of filter parameters, $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$ is the adjacency matrix of the graph augmented with self-loops and $\tilde{\mathbf{D}}$ is the corresponding degree matrix with $\tilde{\mathbf{D}}_{ii} = \sum_{j=1}^{N} \tilde{\mathbf{A}}_{ij}$. This convolved signal matrix constitutes the basic operation (so-called propagation rule) in GCNs.

Likelihood Our likelihood model assumes that, conditioned on all the features and the graph adjacency matrix, the observed labels $\mathbf{Y}^o$ are conditionally independent, i.e.,

$$p_{\boldsymbol{\theta}}(\mathbf{Y}^o \,|\, \mathbf{X}, \mathbf{A}) = \prod_{\mathbf{y}_n \in \mathbf{Y}^o} p_{\boldsymbol{\theta}}(\mathbf{y}_n \,|\, \mathbf{X}, \mathbf{A}) \quad \text{with} \quad p_{\boldsymbol{\theta}}(\mathbf{y}_n \,|\, \mathbf{X}, \mathbf{A}) = \mathrm{Cat}(\mathbf{y}_n \,|\, \boldsymbol{\pi}_n), \tag{1}$$

where $\mathrm{Cat}(\mathbf{y}_n|\boldsymbol{\pi}_n)$ denotes a Categorical distribution over $\mathbf{y}_n$ with parameters $\boldsymbol{\pi}_n$ being the n-th row of the $N \times C$ probability matrix $\boldsymbol{\Pi}$ obtained from a GCN with $L = 2$ layers and ReLU activations,

$$\boldsymbol{\Pi} = \mathbf{f}^{(L)}(\mathbf{X}, \mathbf{A}) = \mathtt{softmax}\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathtt{relu}\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}\mathbf{W}_0\right)\mathbf{W}_1\right), \tag{2}$$

and $\boldsymbol{\theta} = \{\mathbf{W}_0, \mathbf{W}_1\}$ denotes the GCN's weight parameters. One of the fundamental differences of our approach with standard GCNs is that we consider a prior over graphs that is constructed using the observed (but potentially noisy or unreliable) adjacency matrix.

Prior over graphs We consider random graph priors of the form

$$p(\mathbf{A}) = \prod_{ij} p(A_{ij}), \quad \text{with} \quad p(A_{ij}) = \mathrm{Bern}(A_{ij} \,|\, \rho^o_{ij}), \tag{3}$$

where $\mathrm{Bern}(A_{ij} \,|\, \rho^o_{ij})$ is a Bernoulli distribution over A_{ij} with parameter ρ^o_{ij}. Our prior is constructed given an observed (auxiliary) graph $\bar{\mathbf{A}}$ but, for simplicity in the notation, we omit this conditioning

here and in all related distributions. This prior can be constructed in various ways so as to encode our beliefs about the graph structure and about how much this structure should be trusted a priori for our semi-supervised classification task. For example, we have found that the construction $\rho^o_{ij} = \bar{\rho}_1 \bar{\mathbf{A}}_{ij} + \bar{\rho}_0(1 - \bar{\mathbf{A}}_{ij})$, with $0 < \bar{\rho}_1, \bar{\rho}_0 < 1$ being hyperparameters, works very well in practice as it gives just enough flexibility to encode our degree of belief on the absence and presence of links separately. In § 4, we will refer to these hyperparameters as smoothing parameters.

3 Graph structure inference via smooth parameterizations

Our goal is to carry out joint inference over the GCN parameters and the graph structure as given by the adjacency matrix. Since the main additional component of our approach is to consider a prior over the adjacency matrix, we focus on the estimation of the posterior over this matrix given the observed data, i.e. $p(\mathbf{A} \,|\, \mathbf{X}, \mathbf{Y}^o) \propto p_{\boldsymbol{\theta}}(\mathbf{Y}^o \,|\, \mathbf{X}, \mathbf{A}) p(\mathbf{A})$ where the likelihood and prior terms are given by eqs. 1 and 3, respectively. However, computation of this posterior is intractable so we resort to approximate posterior inference methods such as VI [11].

3.1 Variational distribution: free vs smooth parameterizations

Similarly to the prior definition, our approximate posterior is of the form

$$q_{\boldsymbol{\phi}}(\mathbf{A}) = \prod_{ij} q(A_{ij}), \quad \text{with} \quad q_{\boldsymbol{\phi}}(A_{ij}) = \text{Bern}(A_{ij} \,|\, \rho_{ij}), \quad \rho_{ij} > 0, \tag{4}$$

where, henceforth, we use $\boldsymbol{\phi}$ to denote all the parameters of the variational posterior. In the case where ρ_{ij} are free parameters then $\boldsymbol{\phi} = \{\rho_{ij}\}$. We refer to this approach as the 'free' parameterization. Unfortunately, as illustrated in the supplement (appendix D.1), such a parameterization makes optimization of the ELBO wrt $\boldsymbol{\phi}$ extremely difficult and one is forced to use alternative representations of the posterior. Intuitively, conditional independence in the posterior is a strong assumption and small changes in ρ_{ij} will compete with each other to explain the data. Consequently, any continuous optimization algorithm will find it very challenging to find a good direction in this non-smooth combinatorial space. Therefore, we propose the following smooth parameterization:

$$\rho_{ij} = \sigma(\mathbf{z}_i^T \mathbf{z}_j + b_i + b_j + s), \quad \mathbf{z}_i \in \mathbb{R}^d, \quad \{b_i, s \in \mathbb{R}\}, \quad i, j = 1, \dots, N, \tag{5}$$

where $\sigma(x) \equiv (1 + \exp(-x))^{-1}$ is the logistic sigmoid function and $d \leq D$ is the dimensionality of the latent representation $\mathbf{z}$. As we see, the same representation is shared across the columns and rows of $\mathbf{A}$'s Bernoulli parameters, which addresses the combinatorial nature of the optimization landscape of the free parameterization. We note that this parameterization is referred to in the matrix-factorization and link-prediction literature as low-rank [17, 18] or dot-product [14]. In this case the variational parameters are $\boldsymbol{\phi} = \{\{\mathbf{z}_i, b_i\}, s\}$.

3.2 Variational distribution: discrete vs relaxed

We have defined above a variational distribution that naturally models the discrete nature of the adjacency matrix $\mathbf{A}$. Our goal is to estimate the parameters $\boldsymbol{\phi}$ of the posterior $q_{\boldsymbol{\phi}}(\mathbf{A})$ via maximization of the ELBO. For this purpose we can use the score function method [22], which provides an unbiased estimator of the gradient of an expectation of a function using Monte Carlo (MC) samples. However, because of its generality, the score function estimator can suffer from high variance [23].

Therefore, as an alternative to the score-function estimator, we can use the re-parameterization trick [13, 24], which generally exhibits lower variance. Unfortunately, the re-parameterization trick is not applicable to discrete distributions and we need to resort to continuous relaxations. In this work we use Concrete distributions as proposed by [10, 16]. In particular, we denote our binary Concrete posterior distribution with location parameters $\lambda_{ij} > 0$ and temperature $\tau > 0$ as $q_{\boldsymbol{\phi}}(A_{ij}) = \text{BinConcrete}(A_{ij} \,|\, \lambda_{ij}, \tau)$. Analogously, as discussed in [16], in order to maintain a lower bound during variational inference we also relax our prior so that $p(A_{ij}) = \text{BinConcrete}(A_{ij} \,|\, \lambda^o_{ij}, \tau_o)$. In this case the variational parameters are the parameters of the Concrete distribution which can be, as in the discrete case, free parameters $\boldsymbol{\phi} = \{\lambda_{ij}\}$ or have a smooth parameterization analogous to that in eq. 5, i.e. $\lambda_{ij} = \exp(\mathbf{z}_i^T \mathbf{z}_j + b_i + b_j + s)$ and, consequently, $\boldsymbol{\phi} = \{\{\mathbf{z}_i, b_i\}, s\}$.

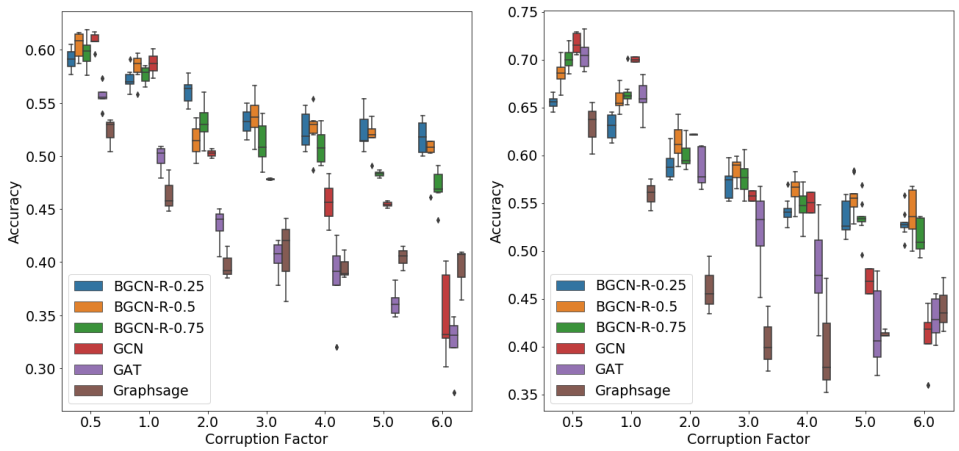


Figure 1: Results for noisy graphs on CITESEER (left) and CORA (right): accuracy as a function of the corruption factor, i.e. the ratio of additional fake links relative to the number of true links. BGCN-R-$\bar{\rho}_1$ stands for the relaxed version of our method with smoothing parameter $\bar{\rho}_1$.

Evidence lower bound The ELBO is given by $\mathcal{L}_{\text{ELBO}}(\boldsymbol{\phi}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{A})} \log p_{\boldsymbol{\theta}}(\mathbf{Y}^o \,|\, \mathbf{X}, \mathbf{A}) - \text{KL}\left[q_{\boldsymbol{\phi}}(\mathbf{A}) \parallel p(\mathbf{A})\right]$, where $\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{A})} \log p_{\boldsymbol{\theta}}(\mathbf{Y}^o \,|\, \mathbf{X}, \mathbf{A})$ is the expected log likelihood (ELL), i.e. the expectation of the conditional likelihood over the approximate posterior, and $\text{KL}\left[q_{\boldsymbol{\phi}}(\mathbf{A}) \parallel p(\mathbf{A})\right]$ is the Kullback-Leibler (KL) divergence between the approximate posterior and the prior. We estimate the variational parameters $\boldsymbol{\phi}$ via gradient-based optimization of the ELBO with the gradients obtained using the re-parameterization trick and automatic differentiation, see appendix B for details.

4 Experiments

In this section we evaluate the performance of our Bayesian approach on the problem of semi-supervised classification using the well-known citation-network datasets CORA and CITESEER. In these datasets, instances (nodes) are documents characterized by features and the class for each document is given by their subject. The required auxiliary (undirected) graph $\bar{\mathbf{A}}$ is obtained based on the citations between the documents. We corrupt the original graph via the insertion of fake links and, consequently, obtain a new $\bar{\mathbf{A}}$. These fake links are added to the graph as a proportion of the number of existing links, which we refer to as corruption factor. We investigate several smoothing parameters $\bar{\rho}_1 = \{0.25, 0.5, 0.75\}$ and a fixed $\bar{\rho}_0 = 10^{-5}$.

Details of the datasets can be found e.g. in [15] and the full description of the experimental set-up in the supplementary material (appendix C). We compare against standard GCN [15], sample-and-aggregate (GRAPHSAGE) [7] and graph attention network (GAT) [26]. [2] Throughout our experiments we use the test accuracy as given by the proportion of correctly classified test examples and the test mean log likelihood (MLL). In both cases, the higher the better.

Accuracy results (MLL values are reported in the supplement) for CITESEER are shown in fig. 1 (left) where we can see that our method, for all the smoothing parameters considered, outperforms the baselines significantly after considering a corruption factor greater than 2. The advantages of our method are less pronounced in CORA (fig. 1, right) but we still see significant improvements for higher noise levels.

5 Conclusion

We have presented a Bayesian approach to graph convolutional networks that addresses their limitations when dealing with noisy graphs. Our inference algorithm for posterior estimation relies on a smooth parameterization of the posterior and has been shown to work very well in real citation graph datasets. An immediate direction for future work is to address the scalability of our method.

[2] Implementations are from https://github.com/stellargraph/stellargraph.

A Binary discrete distributions

The KL divergence between two Bernoulli distributions $q(a \mid \rho)$ and $p(a \mid \rho^o)$ can be computed as

$$\mathrm{KL}\left[q(a \mid \rho) \parallel p(a \mid \rho^o)\right] = \rho[\log \rho - \log \rho^o] + (1-\rho)[\log(1-\rho) - \log(1-\rho^o)]. \tag{6}$$

B Binary concrete distributions

In this section we give details of the re-parameterization used for the implementation of our algorithm when both the prior and the approximate posterior are relaxed via the binary Concrete distribution [10, 16].

B.1 Summary of Bernoulli relaxation transformations

$$A_{ij} \sim \mathrm{BinConcrete}(\lambda_{ij}, \tau) \quad \Leftrightarrow \quad A_{ij} = \sigma(B_{ij}),\ B_{ij} \sim \mathrm{Logistic}\left(\frac{\log \lambda_{ij}}{\tau}, \frac{1}{\tau}\right), \tag{7}$$

$$B_{ij} \sim \mathrm{Logistic}\left(\frac{\log \lambda_{ij}}{\tau}, \frac{1}{\tau}\right) \quad \Leftrightarrow \quad B_{ij} = \frac{\log \lambda_{ij} + L}{\tau},\ L \sim \mathrm{Logistic}(0,1), \tag{8}$$

$$L \sim \mathrm{Logistic}(0,1) \quad \Leftrightarrow \quad L = \sigma^{-1}(U) := \log U - \log(1-U),\ U \sim \mathrm{Uniform}(0,1). \tag{9}$$

In summary, we have

$$A_{ij} \sim \mathrm{BinConcrete}(\lambda_{ij}, \tau) \quad \Leftrightarrow \quad A_{ij} = \sigma\left(\frac{\log \lambda_{ij} + \sigma^{-1}(U)}{\tau}\right),\ U \sim \mathrm{Uniform}(0,1). \tag{10}$$

B.2 Re-parameterized ELBO

With the results above, it is easy to see that we can write the ELBO as:

$$\mathcal{L}_{\mathrm{ELBO}}(\boldsymbol{\phi}) = \mathbb{E}_{q_{\boldsymbol{\phi},\tau}(\mathbf{A})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{Y} \mid \mathbf{X}, \mathbf{A}) - \log \frac{q_{\boldsymbol{\phi},\tau}(\mathbf{A})}{p_{\tau_o}(\mathbf{A})}\right] \tag{11}$$

$$= \mathbb{E}_{g_{\boldsymbol{\phi},\tau}(\mathbf{B})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{Y} \mid \mathbf{X}, \sigma(\mathbf{B})) - \log \frac{q_{\boldsymbol{\phi},\tau}(\sigma(\mathbf{B}))}{p_{\tau_o}(\sigma(\mathbf{B}))}\right] \tag{12}$$

$$= \mathbb{E}_{g_{\boldsymbol{\phi},\tau}(\mathbf{B})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{Y} \mid \mathbf{X}, \sigma(\mathbf{B})) - \log \frac{g_{\boldsymbol{\phi},\tau}(\mathbf{B})}{f_{\tau_o}(\mathbf{B})}\right]. \tag{13}$$

where

$$g_{\boldsymbol{\phi},\tau}(B_{ij}) = \mathrm{Logistic}\left(B_{ij} \mid \frac{\log \lambda_{ij}}{\tau}, \frac{1}{\tau}\right), \quad f_{\tau_o}(B_{ij}) = \mathrm{Logistic}\left(B_{ij} \mid \frac{\log \lambda_{ij}^o}{\tau_o}, \frac{1}{\tau_o}\right). \tag{14}$$

The expectation $E_{g_{\boldsymbol{\phi},\tau}(\mathbf{B})}$ is estimated using S samples from the re-parameterized posterior, which can be obtained using eq. 16 below. Estimation of the variational parameters $\boldsymbol{\phi}$ is done via gradient-based optimization of the ELBO with the gradients obtained by automatic differentiation.

B.3 Predictions

The posterior predictive distribution of the latent labels, given our factorized assumption of the conditional likelihood in eq. 1, can be obtained as

$$p(\mathbf{Y}^u \mid \mathbf{Y}^o, \mathbf{X}) = \sum_{\mathbf{A}} p_{\boldsymbol{\theta}}(\mathbf{Y}^u \mid \mathbf{X}, \mathbf{A}) p(\mathbf{A} \mid \mathbf{Y}^o, \mathbf{X}) \approx \frac{1}{S} \sum_{s=1}^{S} p_{\boldsymbol{\theta}}(\mathbf{Y}^u | \mathbf{X}, \mathbf{A}^{(s)}), \tag{15}$$

where $\mathbf{A}^{(s)}$ is a sample from the posterior $q_{\boldsymbol{\phi}}(\mathbf{A})$, S is the total number of samples and $p_{\boldsymbol{\theta}}(\mathbf{Y}^u \mid \mathbf{X}, \mathbf{A})$ is the GCN-likelihood given in eq. 1. These samples can be obtained as:

$$U \sim \mathrm{Uniform}(0,1), \quad A_{ij}^{(s)} = \sigma\left(\frac{\log \lambda_{ij} + \log U - \log(1-U)}{\tau}\right), \tag{16}$$

where, as before, $\{\lambda_{ij}\}$ are the estimated parameters of the posterior and τ is the posterior temperature.

Table 1: Datasets used in the experiments. Label rate refers to the proportion of labeled nodes used for training.

Dataset	Type	Nodes	Edges	Classes	Features	Label rate
CORA	Citation network	2,708	5,429	7	1,433	0.052
CITESEER	Citation network	3,327	4,732	6	3,703	0.036
TWITTER	Social network	4,971	14,591	2	205	0.01

C Full details of experimental set up

We use two citation-network datasets (CORA and CITESEER) and another real dataset based on Twitter data (TWITTER). In the citation networks the nodes represent documents and their links refer to citations between documents. We construct an undirected graph based on these citation links so as to obtain a binary adjacency matrix. Each document is characterized by a set of features, which are either bag-of-words (BOW) or term frequency-inverse document frequency (TF-IDF) for CORA and CITESEER, respectively. The details of these datasets are shown in table 1. The class for each document is given by their subject and our goal is to predict this for a subset of documents in the network.

TWITTER is based on the dataset published in [25]. Each node represents a Twitter user. An edge between two users exists if a user has re-tweeted the other user. Each node has an associated feature vector such that each feature is derived either from the user's activity or the user's tweet texts. Description of these features is given in the supplement (appendix D.5). The class for each user is given by a binary status property that indicates whether the user is hateful or not; our goal is to predict a user's hatefulness status. We note that the Twitter graph is derived from users' activities and it is not equal to the Twitter social network based on follower relationships. Because of this, we believe this dataset is a good example of a real noisy graph.

For standard GCN and our methods we use a two-layer GCN as given in eq. 2. We train standard GCN exactly as done by [15] so as to minimize the cross-entropy loss on the same datasets, i.e. using dropout, L2 regularization, Glorot weight initialization [6] and row-normalization of input-feature vectors. Hyperparameters were set to those found to perform best by [15], i.e. dropout rate of 0.5, L2 regularization of 5×10^{-4} and 16 hidden units. The ADAM optimizer [12] was used with an initial learning rate of 0.01.

For our method, we carry out point estimation of the GCN-likelihood parameters and the approximate posterior parameters so as to maximize the ELBO. Hyperparameters for GCN-likelihood estimation were the same as above. To construct the prior over the adjacency matrix we followed the procedure explained in § 2 with various smoothing parameters $\bar{\rho}_1$ and $\bar{\rho}_0 = 10^{-5}$. Posterior initialization was done to be as close as possible to the prior by setting the latent variables $\mathbf{Z}$ via low-rank factorization with the number of latent dimensions set to $d = 1,000$. The biases $\{b_i\}$ and shift s in eq. 5 were initialized to 0 and -20, respectively. Prior and posterior temperatures τ_o and τ were set to 0.1. All the methods were executed up to a maximum of $5,000$ epochs (unless otherwise stated explicitly) with $S = 3$ samples used for estimating the required expectations.

We used our own implementation for GAT and GRAPHSAGE. GAT used an architecture identical to the one described in [26]. The first layer consists of $K = 8$ attention heads computing $F = 8$ features, followed by an exponential linear unit (ELU) nonlinearity. The second layer is used for classification that computes C features (where C is the number of classes), followed by a softmax activation. $L2$ regularization with $\lambda = 0.0005$ and 0.6 dropout was used. The implementation of GRAPHSAGE used mean aggregator functions and sampled the neighborhood at a depth of $K = 2$ with neighborhood sample size of $S_1 = S_2 = 32$. The model was trained with 0.5 dropout and $L2$ regularization with $\lambda = 0.0005$.

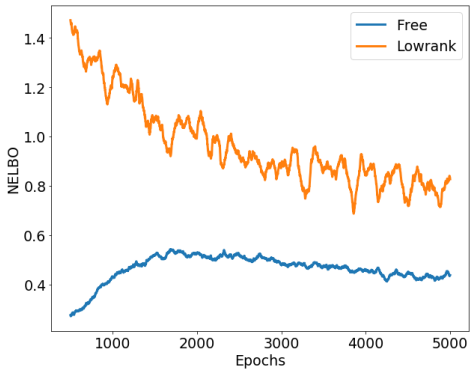


Figure 2: The negative evidence lower bound (NELBO) as a function of the number of epochs for the freely parameterized posterior (Free) and the smooth parameterization (Lowrank).

Table 2: Results for noiseless graphs: 'no graph' corresponds to ignoring the input graph and 'full graph' corresponds to using the full original graph without corruption. Graph convolutional network (GCN); Discrete Bayesian graph convolutional network (BGCN-D) and Relaxed Bayesian graph convolutional network (BGCN-R). Accuracy (ACC) and mean log likelihood (MLL).

	CORA- no graph				CORA- full graph			
Method	ACC	(std dev)	MLL	(std dev)	ACC	(std dev)	MLL	(std dev)
GCN	0.5592	0.0119	-1.3058	0.0309	0.8178	0.0041	-0.6287	0.0107
BGCN-D	0.5494	0.0084	-1.3290	0.0222	0.8049	0.0024	-0.6760	0.0092
BGCN-R	0.5584	0.0102	-1.2985	0.0257	0.8193	0.0045	-0.6306	0.0076
	CITESEER- no graph				CITESEER- full graph			
GCN	0.5537	0.0151	-1.2474	0.0166	0.7041	0.0068	-0.9342	0.0075
BGCN-D	0.5484	0.0082	-1.2654	0.0105	0.6837	0.0053	-1.0351	0.0222
BGCN-R	0.5491	0.0071	-1.2631	0.0122	0.7056	0.0041	-0.9418	0.0072

D Additional results

D.1 Smooth parameterization

Figure 2 compares a typical run of our algorithms using the free and smooth parameterizations of the posterior. We see that the optimizer struggles to find directions of improvement for the free parameterization, whereas for the smooth parameterization the negative evidence lower bound (NELBO) decreases steadily during optimization. This behavior was observed throughout several learning rate settings and we attribute it to the discrete combinatorial optimization nature of the NELBO when using the free parameterization.

D.2 Noiseless graphs

Table 2 shows the results for the noisy graphs where we see that our approach performs similarly to standard GCNs.

D.3 Noisy graphs

Test mean log likelihood (MLL) results are given in fig. 3.

D.4 Feature-based graphs

Following the methods of [9], a graph was estimated using CORA and CITESEER based on the node features. Given a training set with features and labels we train a fully connected network with 1-hidden layer, $Q \in \{16, 32\}$ neurons using standard rectified linear unit (RELU) activation and

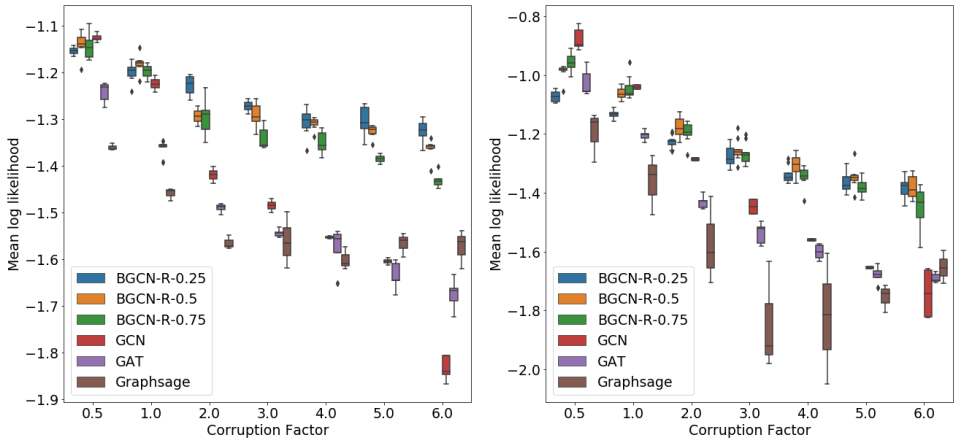


Figure 3: Results for noisy graphs on CITESEER (left) and CORA (right): Mean log likelihood (MLL) as a function of the ratio of additional fake links relative to the number of true links. BGCN-R-$\bar{\rho}_1$ stands for the relaxed version of our method with smoothing factor $\bar{\rho}_1$.

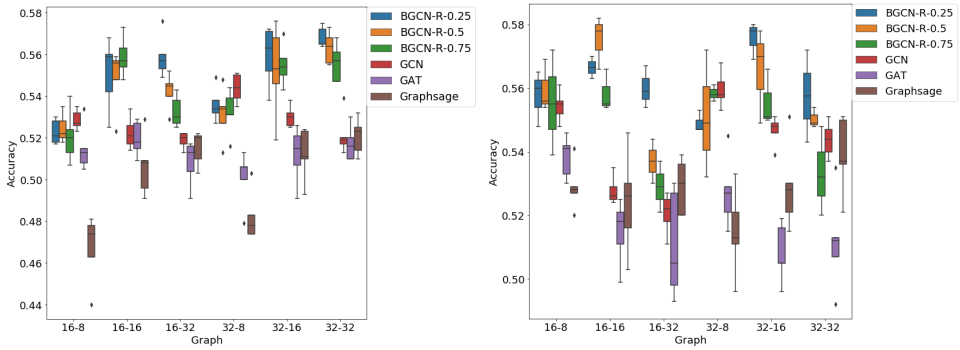


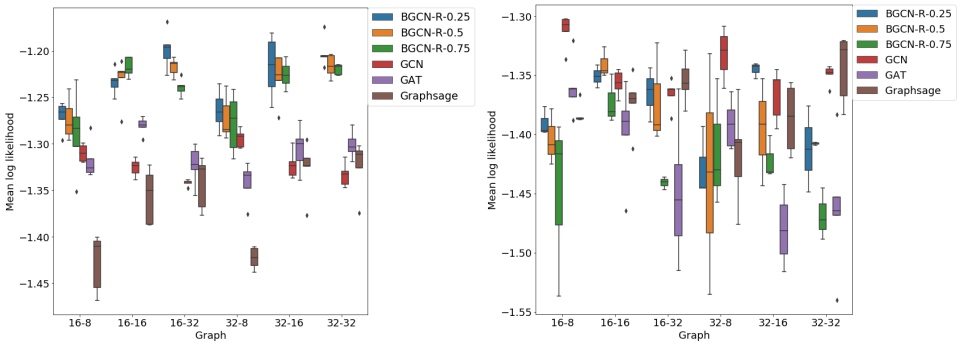
Figure 4: Results for feature-based graphs on CITESEER (left) and CORA (right): Accuracy as a function of Q-K settings, where Q is the number of hidden units and K the number of neighbors used to construct the graph. BGCN-R-$\bar{\rho}_1$ stands for the relaxed version of our method with smoothing factor $\bar{\rho}_1$.

0.5 dropout between each layer. We then extract the first layer features $\tilde{\mathbf{Z}}_1 \in \mathbb{R}^{M \times Q}$ where M includes training, validation and test sets and consider their distance, $d(i,j) = \| \tilde{\mathbf{Z}}_{1,i} - \tilde{\mathbf{Z}}_{1,j} \|^2$. The adjacency matrix, $\mathbf{A}$, was constructed by taking the $K \in \{8, 16, 32\}$ closest neighbors. Test mean log likelihood (MLL) results are given in figs. 4 and 5.

D.5 Twitter

The TWITTER dataset we use here is a subset of that published in [25][3]. We modified the original dataset in two ways. First, the graph in [25] has $100,000$ nodes of which only $4,971$ have manually been annotated as hateful or not. For our experiments we consider the subgraph of the annotated nodes only. The original graph has approximately $2,200,000$ edges whereas the annotated users subgraph has $14,591$ edges. Second, we use a subset of the node features in [25]. The latter includes manually engineered graph features such as several node centrality measures. Furthermore, [25] includes features derived from the lexical analysis of the users' tweets based on two different text representation learning methods, Glove [21] and Empath [4]. We have removed all the graph-based manually engineered features. Lastly, we only consider the Empath lexical features and ignore the Glove ones since we expect them to encode similar information about the content of a user's tweets.

[3]This dataset can be downloaded from Kaggle at https://www.kaggle.com/manoelribeiro/hateful-users-on-twitter



1

Figure 5: Results for feature-based graphs on CITESEER (left) and CORA (right): Mean log likelihood (MLL) as a function of Q-K settings, where Q is the number of hidden units and K the number of neighbors used to construct the graph. BGCN-R-$\bar{\rho}_1$ stands for the relaxed version of our method with smoothing factor $\bar{\rho}_1$.

The nodes in the TWITTER dataset for our empirical study have feature vectors with 205 dimensions. [4]

We split the data into training, validation, and test sets as follows. We use 25 labeled examples from each class for training such that the size of the training set is 50 or only 1% of the total number of nodes in the graph. We then split the remaining nodes in to validation and test sets such that the validation set contains 30% of the data and the test set the remaining 70%. All data are sampled uniformly at random. We generated 20 such splits of the data and used them consistently in the evaluation.

We compare 4 Bayesian GCN models, the relaxed version, using different values for the edge smoothing parameter and prior, posterior temperatures against a GCN and an MLP model. All BGCN, GCN, and MLP models use a single hidden layer with 16 hidden units. The experimental setup is as described here in appendix C and in the main paper § 4 unless stated otherwise.

The results are shown in Figure 6. For our method the notation BGCN-$\bar{\rho}_1$-Pr-Po indicates that $\bar{\rho}_1$ is the smoothing factor for the existing edges in the graph; Pr is the temperature used for the prior distribution; and Po is the temperature used for the posterior distribution. We trained all models for a maximum 5000 epochs and used the accuracy on the validation set for early stopping. We report the accuracy and mean log likelihood on the test set for the latter model. Each model was trained and evaluated on each of the 20 splits of the data. The results in Figure 6 indicate that all models exhibit high predictive variance. Generally, BGCN models with a high degree of smoothing and temperature parameters in the range suggested in [16], show a trend towards better performance with regards to the mean or median values for each set of experiments. However, we emphasize that the high predictive variance of all models indicates that we cannot say that any model is statistically significantly better than the rest.

References

[1] Edoardo M Airoldi, David M Blei, Stephen E Fienberg, and Eric P Xing. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*, 9(Sep):1981–2014, 2008.

[2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations*, 2014.

[3] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016.

[4] We will be releasing this dataset together with the source code implementing our variational spectral graph convolutional network algorithm.

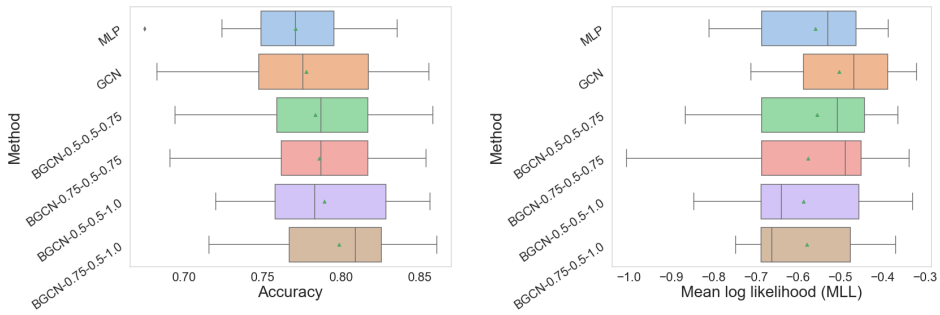


Figure 6: Results for TWITTER: Box plots for machine learning model vs accuracy (left) and mean log likelihood (right) on the TWITTER test set. BGCN-$\bar{\rho}_1$-Pr-Po indicates that $\bar{\rho}_1$ is the smoothing factor for the existing edges in the graph; Pr is the temperature used for the prior distribution; and Po is the temperature used for the posterior distribution. For each box plot, the vertical lines indicate median values and the triangles indicate the mean values of the data. Outliers are shown using diamonds.

[4] Ethan Fast, Binbin Chen, and Michael S Bernstein. Empath: Understanding topic signals in large-scale text. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 4647–4657. ACM, 2016.

[5] Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. Learning discrete structures for graph neural networks. In *International Conference on Machine Learning*, 2019.

[6] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Artificial Intelligence and Statistics*, pages 249–256, 2010.

[7] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[8] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on Graphs via Spectral Graph Theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[9] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[10] Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparameterization with Gumbel-Softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[11] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.

[14] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[15] Thomas N Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*, 2017.

[16] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. *arXiv preprint arXiv:1611.00712*, 2016.

[17] Aditya Krishna Menon and Charles Elkan. Link prediction via matrix factorization. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 437–452. Springer, 2011.

[18] Andriy Mnih and Ruslan R Salakhutdinov. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, pages 1257–1264, 2008.

[19] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5115–5124, 2017.

[20] Yin Cheng Ng, Nicolò Colombo, and Ricardo Silva. Bayesian semi-supervised learning with graph Gaussian processes. In *Advances in Neural Information Processing Systems*, pages 1683–1694, 2018.

[21] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[22] Rajesh Ranganath, Sean Gerrish, and David M. Blei. Black box variational inference. In *Artificial Intelligence and Statistics*, 2014.

[23] Rajesh Ranganath, Dustin Tran, and David Blei. Hierarchical variational models. In *International Conference on Machine Learning*, pages 324–333, 2016.

[24] Danilo J Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, 2014.

[25] Manoel Horta Ribeiro, Pedro H Calais, Yuri A Santos, Virgílio AF Almeida, and Wagner Meira Jr. "Like sheep among wolves": Characterizing hateful users on Twitter. *arXiv preprint arXiv:1801.00317*, 2017.

[26] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *International Conference on Learning Representations*, 2018.

[27] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019.

[28] Yingxue Zhang, Soumyasundar Pal, Mark Coates, and Deniz Üstebay. Bayesian graph convolutional neural networks for semi-supervised classification. In *AAAI Conference on Artificial Intelligence*, 2019.

[29] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.