GNN Interpretability Using Bayesian Inference

Shalin Patel[1,2]

[1] *Division of Applied Mathematics, Brown University*
[2] *Department of Computer Science, Brown University*

Abstract

1 Introduction

Graphs serve as a natural repository for information in many real-world applications ranging from social, informational, chemical, and biological domains [1]. Especially as data becomes more and more unstructured, graphs represent a flexible manner for storing and relating different nodes and their related features [2]. Indeed, graphs represent one of the most general mathematical structures for relating data and are seeing increasing use in modeling phenomenon such as social networks and gene regulatory networks [2, 3]. For the purposes of this work, given a set of vertices V, node features $\mathcal{X} : V \to \mathbb{R}^d$, a set of edges $E \subseteq V \times V$, and weights on the edges $W : E \to [0, 1]$, we consider the graph $G = \{V, \mathcal{X}, E, W\}$. Additionally, we let the space of all graphs for a given set of vertices V be $\mathcal{G}$. Below in figure 1, a simple visualization of this definition can be seen for a cyclic graph of order 3.

1.1 Graph Neural Networks

To deal with the proliferation of graphs in computing and the need to construct models that consider graphs as a first-class member of the modeling process, a class of models known as

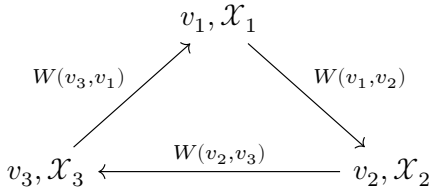

Figure 1: An example graph as related to the terminonlogy laid out above

Graph Neural Networks have emerged (GNN) with state of the art performance on a variety of classification and regression tasks [4]. Specifically, GNNs and their early iterations in GCNs took inspiration from CNNs that represented applying successive convolution operations on regular grids of information, such as images, to compose local features in the grid in to higher-level predictions [5]. At a high level, most GNN frameworks can be split into three steps `MSG`, `AGG`, and `UPD` representing a messaging step, aggregation step, and update step, respectively.

At a layer l in a GNN model ϕ, the update of the hidden state of the model occurs first by sending messages for all $(v_i, v_j) \in E$ as a function of the hidden state $\mathcal{H}_i^{l-1}$ and $\mathcal{H}_j^{l-1}$ as well as the weight $W_{ij} := W(v_i, v_j)$. Specifically, we have

$$m_{ij}^l := \texttt{MSG}(\mathcal{H}_i^{l-1}, \mathcal{H}_j^{l-1}, W_{ij})$$

Then, a GNN performs an aggregation step wherein it calculates an aggregate message for every vertex $v \in V$. Let $\mathcal{N}_k : V \times E \to \mathcal{P}(E)$ be a function that returns the edges in the k-hop neighborhood of a node. Then, we can formally write the aggregation step as

$$M_i^l := \texttt{AGG}(\{m_{ij}^l \mid v_j \in \mathcal{N}_k(v_i)\})$$

Then, finally, at each node, the GNN takes a nonlinear function (often a neural network of some sort) and applies it to this aggregated message M_i^l along with the hidden state $\mathcal{H}_i^{l-1}$ to get the new hidden state.

$$\mathcal{H}_i^l := \texttt{UPD}(\mathcal{H}_i^{l-1}, M_i^l)$$

When composed in layers, this forms a full Graph Neural Network. Note that in this framework, $\mathcal{H}_i^0 := \mathcal{X}_i$. Based on the task type, either node or graph classification in this work, further layers may be added on top of the final output. For example, in graph classification, it is often the case that the final node embeddings are concatenated and then run through an MLP to get a final classification for the whole graph [4].

The specific layers used in this work are Graph Convolution Layers also known as GCNs [6]. In the context of the framework above, the `MSG` sends weighted and normalized node features from the $l-1$st layer where the normalization is by the out-degree of the sending node and the weight of the message coming from $\mathcal{W}_{ij}$. In the `AGG` step, all of these messages are summed together. Finally, the `UPD` step applies a neural network, usually a trainable linear layer combined with a non-linear activation function to provide a non-linear update step.

1.2 Interpretation on GNNs

Given a GNN, it is natural in many fields such as computational biology to perform interpretation on the model in order to gain further insights. For example, in the case of RNA-seq data, a natural question is to determine important regulatory pathways between

genes that could be related to eventual up or down regulation of a target gene [3]. One natural way to perform this task is to train a GNN on the RNA-seq data for a node-classification task and feed it a large graph G with many redundant edges. Given a GNN model ϕ with l layers and a target gene $v_i \in V$, we would like to determine $\mathcal{E}_i \subseteq \mathcal{N}_l(V_i, E)$ as well as $\mathcal{W}_i : \mathcal{E}_i \to [0, 1]$ such that $\mathcal{E}_i, \mathcal{W}_i$ represent the most important subgraph for the model ϕ to perform its predictions for the input vertex V_i. While there are a few criteria for determining importance, we consider importance to be the maximal mutual information between the original model with its inputs and the same model with the estimated subgraph $\mathcal{E}_i, \mathcal{W}_i$. Written more formally, we wish to discover

$$\arg\max_{\mathcal{E}_i, \mathcal{W}_i} \texttt{MI}(\phi(V_i, \mathcal{X}, E, W), \phi(V_i, \mathcal{X}, \mathcal{E}_i, \mathcal{W}_i))$$

This is a computationally hard problem as a brute force search would take $O(2^{|E|})$ time even when discounting the weight array $\mathcal{W}_i$. In practice, discovering $\mathcal{E}_i$ is ignored and most of the importance discovery is done through learning a suitable $\mathcal{W}_i$ while letting $\mathcal{E}_i = E$.

However, because of the given task, and the various properties we would like to see in a reasonable interpretation of a GNN model, there are many routes that have been taken to interpret these models. As will be shown in the related works section, there have been a few non-bayesian attempts to solve this problem. The goal of this work is to analyze these methods and then suggest a new Bayesian method to solving the issue of searching for the best subgraph for a trained GNN to perform post-hoc analysis of importance. Furthermore, an added benefit of utilizing a Bayesian approach is that a full conditional distribution over the importance graph is learned giving researchers an even larger level of insight into their model that goes beyond just a simple edge mask.

1.3 Bayesian Inference

Recently, in deep learning literature there has been a rise in utilizing Bayesian methods to create Bayesnian Neural Networks in order to provide uncertainty aware predictions [7]. While primarily concerned with giving estimates for failure modes and reducing overfitting within deep learning methods, the synthesis of Bayesian methods and deep learning models has imbued them with a greater sesne of interpretability and introspectability. In the concext of GNN interpretation, modelling the subgraph $\mathcal{E}_i, \mathcal{W}_i$ as a joint distribution allows for conditionining on certain edge weights and imbues the interpretations that are derived with a greater sense of introspectability. Figure 2 gives a good overview of the deep learning analogues for point estimate neural networks. In the same way, analogues will be utilized in the interpretation task in order to get the same benefits that have already been outlined in BNNs. In the Bayesian paradigm, a distribution $\mathcal{P}$ is treated as the belief in the occurance of a given event from the distribution rather than the limit of the frequencies of each event as in the frequentist scheme. Furthermore, prior beliefs are thought to inform posterior beliefs. In the context of interprebility this is important as the belief for a given

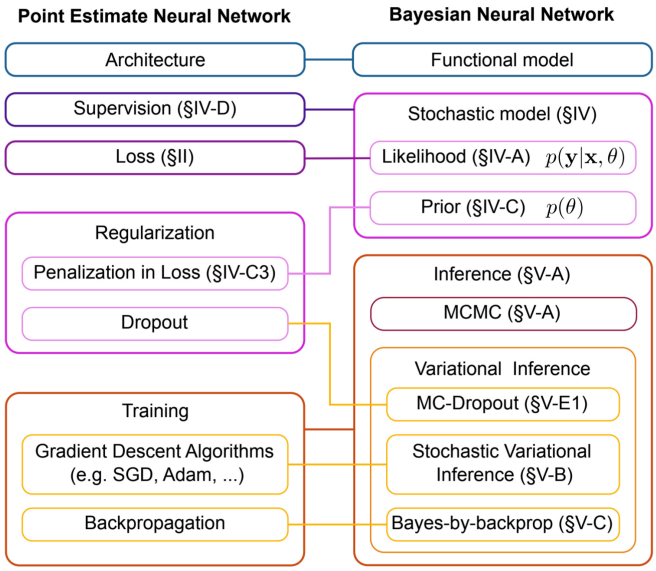

Figure 2: An overview of the corresponsing structures in a standard neural network and a bayesian neural network

interpretaiton is dependant on the domain that it is brought up in. For example, in social networks one may expect relatively dense explanations while in biology they would tend to be sparse [1] [3]. Generally speaking, given a hypothesis H representing the prior belief for the state of a system and some data D, the posterior probability $\mathcal{P}(H \mid D)$ can be calculated as

$$\mathcal{P}(H \mid D) = \frac{\mathcal{P}(D \mid H)\mathcal{P}(H)}{\mathcal{P}(D)}$$

and in this way, the posterior is conditioned by both the prior belief and the evidence that the data presents. While there are a variety of different techniques that can be used to update the prior belief into a posterior distribution as seen in figure 2, in this paper only stochastic variational inference (SVI) and Bayes-by-backprop are utilized.

1.3.1 Stochastic Variational Inference (SVI)

While other inference methods such as MCMC (with popular algorithms including HMC and NUTS [8] [9]), allow exact sampling from the posterior distribution, these methods have proven unpopular with the BNN community due to their algorithmic complexity and lack of scalability to larger models. Hence many communities use SVI which is not an exact method. In SVI, there is a family of distributions $q_\phi(H)$ which are parametrized by parameters ϕ. A common example would be the family of normal distributions parametrized

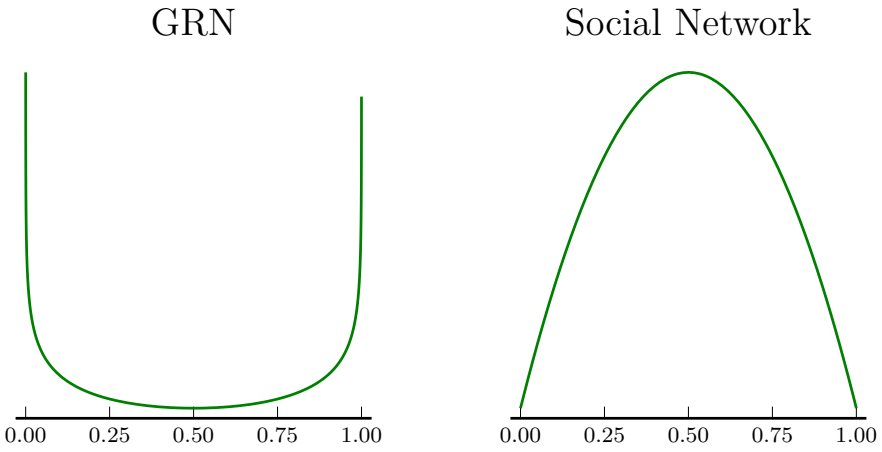

Figure 3: Idealized prior distributions for GRNs and Social Networks based on the Beta distribution. The displayed distributions are $\mathrm{B}(0.95, 0.95)$ and $\mathrm{B}(2, 2)$ respectively

by their mean and covariance structure. The goal of SVI is to approximate the posterior $\mathcal{P}(H \mid D)$ as closely as possible by $q_\phi(H)$. The most common measure of approximation in probability space is given by the Kullback-Leibler divergence (KL-divergence). While not a proper metric over the space of distributions, it does give a computationally-reasonable method to optimize against ϕ to get as close a match as possible. Specfically, SVI aims to minimize

$$D_{KL}(q_\phi \mid\mid \mathcal{P}) = \int_H q_\phi(H') \log \frac{q_\phi(H')}{\mathcal{P}(H' \mid D)} dH'$$

This is still problematic since the quantity $\mathcal{P}(H \mid D)$ would still need to be calculated. Hence, it is sufficient to optimize against the ELBO which serves as a lower-bound for the KL-divergence. The ELBO is defined as

$$\log \mathcal{P}(D) - D_{KL}(q_\phi \mid\mid \mathcal{P}) = \int_H q_\phi(H') \log \frac{\mathcal{P}(H', D)}{q_\phi(H')} dH'$$

Note here that $\log \mathcal{P}(D)$ is just a constant meaning that minimizing the KL-divergence is the same as maximizing the ELBO. Note that, generally speaking, the families q_ϕ tend to come from the exponential family of distributions and the parameters for these families are then just optimized using a typical SGD algorithm such as ADAM [10].

1.3.2 Bayes-by-backprop

While SVI provides a good framework for Bayesian inference, it does not quite work for deep learning applications because stochastiicity stops backpropogation from going through

a neural network. To mitigate this problem, the usual reparametrization technique used in creating variational autoencoders (VAEs) [11] is combined with SVI to create a deep-learning friendly SVI algorithm.

2 Related Work

3 Methods

4 Experimental Setup

5 Results

6 Discussion

References

[1] Eunjoon Cho, Seth A. Myers, and Jure Leskovec. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 1082–1090. Association for Computing Machinery.

[2] Takashi Washio and Hiroshi Motoda. State of the art of graph-based data mining. 5(1):59–68.

[3] Francesca Petralia, Won-Min Song, Zhidong Tu, and Pei Wang. New method for joint network analysis reveals common and different coexpression patterns among genes and proteins in breast cancer. 15(3):743–754.

[4] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. GNNExplainer: Generating explanations for graph neural networks.

[5] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering.

[6] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks.

[7] Laurent Valentin Jospin, Wray Buntine, Farid Boussaid, Hamid Laga, and Mohammed Bennamoun. Hands-on bayesian neural networks – a tutorial for deep learning users. 17(2):29–48.

[8] Michael Betancourt. A conceptual introduction to hamiltonian monte carlo.

[9] Matthew D. Hoffman and Andrew Gelman. The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo.

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization.

[11] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. 12(4):307–392.