Graph Neural Networks with a Distribution of Parametrized Graphs

See Hian Lee [*1] Feng Ji [*1] Kelin Xia [2] Wee Peng Tay [1]

Abstract

Traditionally, graph neural networks have been trained using a single observed graph. However, the observed graph represents only one possible realization. In many applications, the graph may encounter uncertainties, such as having erroneous or missing edges, as well as edge weights that provide little informative value. To address these challenges and capture additional information previously absent in the observed graph, we introduce latent variables to parameterize and generate multiple graphs. The parameters follow an unknown distribution to be estimated. We propose a formulation in terms of maximum likelihood estimation of the network parameters. Therefore, it is possible to devise an algorithm based on Expectation-Maximization (EM). Specifically, we iteratively determine the distribution of the graphs using a Markov Chain Monte Carlo (MCMC) method, incorporating the principles of PAC-Bayesian theory. Numerical experiments demonstrate improvements in performance against baseline models on node classification for both heterogeneous and homogeneous graphs.

1. Introduction

Graph Neural Networks (GNNs) have facilitated graph representational learning by building upon Graph Signal Processing (GSP) principles and expanding their application in the domains of machine learning. Moreover, GNNs have demonstrated their effectiveness across a wide range of tasks in domains such as chemistry (Gilmer et al., 2017a), recommendation systems (Ying et al., 2018; Chen et al., 2022), financial systems (Sawhney et al., 2021) and e-commerce settings (Liu et al., 2022), among others. However, GSP and many GNNs rely on a fixed graph shift operator, such as the adjacency or Laplacian matrix, to analyze and learn from graph data, assuming that the given graph is accurate and noise-free. This approach has inherent limitations, considering that graph data is often uncertain. There are also models such as GTN (Yun et al., 2019) and GAT (Veličković et al., 2018) that determine weighted graph structures or shift operations in the training phase from multiple possibilities. However, a single neighborhood feature aggregation mechanism is usually learned and used for testing.

The uncertainty mentioned above stems from the existence of multiple potential variations in graph constructions as a universal optimal method does not exist. Structural noise, which includes missing or spurious edges, and the absence of informative edge weights, further contributes to the uncertainty in graph data (Zhang et al., 2019; Dong & Kluger, 2023). Handling this uncertainty is crucial as the graph directly influences the results of both GSP and GNNs (Li et al., 2021c).

Several GNN works have recognized that the provided graph in benchmark datasets is suboptimal. For example, in (Topping et al., 2022), a method was introduced to enhance the provided graph by rewiring it at graph bottlenecks. Similarly, in (Li et al., 2021a) and (Ye et al., 2020), approaches were developed to reweigh edges, to reduce information flow at cluster boundaries. Another perspective involves considering the given or observed graph as a particular realization of a graph model, as discussed in (Zhang et al., 2019) (cf. Appendix F on related works). In their work, a Bayesian framework was adopted to learn a more robust model that can withstand perturbations in graph topology. These collective efforts underscore the common observation that the observed graph is often imperfect, and determining the optimal graph is a non-trivial task, as it depends on both the physical connections and the edge weights, which regulate the rates of information transmission (Ji et al., 2023d).

Our work aligns with the viewpoint presented in (Zhang et al., 2019). We conceptualize the observed graph as an individual instance originating from a distribution of graphs, which is influenced by one or more latent parameters. Nevertheless, in contrast to (Zhang et al., 2019) which proposed a Bayesian framework, we propose an EM framework for graph learning and name our model EMGNN. Even though

[*]Equal contribution [1]School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore [2]School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore. Correspondence to: Wee Peng Tay <wptay@ntu.edu.sg>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

both are probabilistic frameworks, the focus is distinctly different. In the case of the Bayesian framework of (Zhang et al., 2019), the focus is on estimating the posterior distribution of model parameters given the data. As such, model parameters are deemed as random variables trained by a series of characteristically similar graphs. Meanwhile, in our EM framework, we seek to maximize the log-likelihood of the observed data in conjunction with the latent variables. Additionally, we permit the generated graphs to demonstrate more pronounced variations. Our main contributions are:

- We present a general framework for modeling the distribution of graphs to handle uncertainty in graph data. The learned distribution provides valuable insights into our model's behavior.
- We formulate the graph learning problem as a maximum likelihood estimation (MLE) so that tools from statistical learning can be applied. The new objective subsumes the classical objective of minimizing empirical loss if the graph is deterministic.
- We evaluate our model on nine datasets in two distinct applications, and observe promising performance compared to the respective baseline methods.
- We inspect the learned graph distribution, confirming that it effectively captures the intricacies of heterogeneous graph datasets, thus validating the utility of our model and framework.

Notations are in Appendix A and proofs are in Appendix G.

2. Preliminaries

2.1. Graph Neural Networks

Graph neural networks (Chen et al., 2020; Kang et al., 2023; Brody et al., 2022; Lee et al., 2021; Zhao et al., 2023), which are neural networks designed to operate on graphs, typically employ the message-passing framework. Within this framework, the features of each node are updated by integrating with those of its neighboring nodes.

More specifically, suppose that we have a graph $G = (V, E)$, where V is the set of vertices and E is the set of edges. Moreover, each node $v \in V$ is associated with (initial) node features represented by x_v^0. The node features can then be updated in the k-th layer as follows:

$$x_v^k = \sigma\big(W^k \text{AGGR}(\{x_v^{k-1} \mid v \in \mathcal{N}(v)\})\big) \tag{1}$$

where σ is an activation function, W^k are the learnable weights in the k-th layer and $\mathcal{N}(v)$ is the set of neighbors of v. AGGR is a message aggregation function. The choice of AGGR defines various variants of GNNs (Xu et al., 2019). For example, the mean operator yields Graph Convolutional Networks (GCN) (Kipf & Welling, 2017), while using the attention mechanism results in Graph Attention Networks (GAT) (Veličković et al., 2018).

In a GNN with K layers, the last layer outputs features $\{x_1^K, \ldots, x_{|V|}^K\}$. For node classification, these features can be used directly. Meanwhile, for a graph-level task, a READOUT graph-level pooling function is needed to obtain the graph-level representation (Xu et al., 2019).

2.2. Signal Processing over a Distribution of Graphs

GNN is closely tied to GSP's theory (Shuman et al., 2013). Briefly, given an undirected graph G, we consider a fixed graph shift operator S such as its adjacency or Laplacian matrix. A graph signal is a vector $\mathbf{x} = (x_v)_{v \in V}$ that associates a number x_v to each node $v \in V$. Intuitively, applying the linear transformation S to $\mathbf{x}$ is considered as a "shift" of $\mathbf{x}$. If S is the normalized adjacency matrix, then it amounts to the AGGR step of (1) for GCN. More generally, if $P(\cdot)$ is a single variable polynomial, then plugging in S results in the matrix $P(S)$, which is called a *convolution filter* in GSP. This notion of convolution appears in (Michael Defferrard, 2016), and has been widely used since then.

On the signal processing side, (Ji et al., 2023d) has developed a theory that generalizes traditional GSP. The authors propose a signal processing framework assuming a family of graphs are considered simultaneously, to tackle uncertainties in graph constructions. Formally, it considers a distribution μ of graph shift operators S_λ parametrized by λ in a sample space Λ. The work develops corresponding signal processing notions such as Fourier transform, filtering, and sampling. In particular, a convolution takes the form $\mathbb{E}_{\lambda \sim \mu}[P_\lambda(S_\lambda)]$, where $P_\lambda(\cdot)$ is a polynomial and $P_\lambda(S_\lambda)$ is an ordinary convolution with shift S_λ. Our work is based on the idea that replaces $P_\lambda(S_\lambda)$ with a more general filter such as a GNN model. As a preview, unlike (Ji et al., 2023d), we introduce an EM framework that simultaneously estimates model parameters and the distribution μ.

3. The Problem Formulation

3.1. Distributions for Different Graph Types

Here, we outline how Λ, a parameter sample space, arises for different graph types, showcasing why the proposed framework is useful for graph-related tasks. Parameters $\lambda \in \Lambda$ can be scalars, vectors, or more general forms, enabling task-specific graph parameterization and diverse graph generation. For instance, a specific λ can indicate the edge weight for an edge type in heterogeneous graphs and be employed to create varied weighted graphs. Details are task-specific and provided in Section 5. Though the schemes are simple and intuitive, there may be alternatives for Λ based

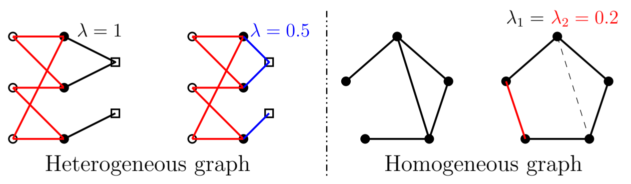


Figure 1. For a heterogenous graph, we may use a parameter λ to control the information transmission rate for each edge type. For example, choosing $\lambda = 1$ or $\lambda = 0.5$ for the edge type between "disc" and "square" nodes yields different weighted graphs. Conversely, in a homogeneous example with 5 initial edges, by choosing $\lambda_1 = \lambda_2 = 0.2$, 20% of the initial and missing edges are randomly removed and added, potentially forming a "pentagon".

on other factors.

Heterogeneous graphs are graph structures characterized by the presence of multiple node types and multiple edge types, imparting a greater degree of complexity compared to *homogeneous graphs*, which consist of a single node and edge type. For a heterogeneous graph, the insight is that we assign a parameter to each edge type, whose distribution is to be estimated and used. Intuitively, in a model based on message passing, the parameters for different edge types are interpreted as different information transmission rates. Such information is not observed in the given graph.

It is less obvious how Λ can be constructed for a homogenous graph. Intuitively, we assume that the observed graph contains "noisy" edges and has missing edges (cf. Zhang et al. (2019)). Therefore, parameters that are interpreted as probabilities for adding and removing edges from the observed graph, can be introduced (see an example in Fig. 1).

Though we have different setups, a unified framework to deal with both is proposed in the next section. To emphasize the importance of having the correct μ on Λ, we note the following result. Details are in Appendix G.1 to avoid overloading the discussion with terms not used in the sequel.

Theorem 1. *(Informal) If the parameterization $\lambda \in \Lambda \mapsto S_\lambda$ is sufficiently continuous, then the μ-expected feature representation of a GNN model, whose layers are of the form (1), changes continuously as the distribution μ varies.*

Discussion: Intuitively, the result claims that if the parameterization is sufficiently regular, then similar distributions on Λ yield GNN models with similar feature representations. Hence, suppose there is a true graph G_0 parameterized λ_0, whose associated GNN model gives a good feature representation. Instead of using a possibly "noisy" observed graph G parameterized by λ, it might be beneficial to use a distribution μ on Λ closer than δ_λ to δ_{λ_0}, even if μ is not a delta distribution. It is less restrictive to allow non-delta graph distributions. Moreover, we are also motivated by the insight that there might be several node connections that contribute with different importance to a learning task.

3.2. Maximum Likelihood Estimation

Motivated by the previous subsection, we consider a distribution μ on a parameter (sample) space $\Lambda \subset \mathbb{R}^r$ of graphs $\{G_\lambda, \lambda \in \Lambda\}$, with a fixed set of nodes V. The space Λ can be finite, countably infinite, or even uncountable. For each G_λ, there is a corresponding shift operator S_λ. We usually assume that μ has a density function $p(\cdot)$ w.r.t. a base measure on Λ. For example, if Λ is finite, we can use the discrete counting measure as the base measure. On the other hand, if Λ is a compact interval in $\mathbb{R}$, then we can choose the Lebesgue measure as the base measure.

Assume that each node $v \in V$ is associated with features x_v. They are collectively denoted by $\mathbf{x}$. Our framework depends on a *fixed GNN model architecture* Ψ, e.g., GCN. It outputs the learned embeddings $\mathbf{z} = \Psi(\lambda, \mathbf{x}; \boldsymbol{\theta})$ given the node features $\mathbf{x}$, the graph parameter λ, and the GNN model parameters $\boldsymbol{\theta}$. These in turn are used to determine a vector of labels $\hat{\mathbf{y}}$. For a task-specific loss $\ell(\cdot,\cdot)$ that compares predicted $\hat{\mathbf{y}}$ and true label (vector) $\mathbf{y}$, we may compute $L_{\mathbf{X}}(\lambda, \boldsymbol{\theta}) = \ell(\hat{\mathbf{y}}, \mathbf{y})$. We use $\mathbf{X}$ to denote the full information $\{\mathbf{x}, \mathbf{y}\}$. We interpret $\mathbf{X}$ as a sample from a random variable, denoted by $\mathfrak{X}$, of collective information of features and labels.

An example of Ψ is the model described by (1). We may also allow parameters $\boldsymbol{\theta}$ and λ to determine W^k. For example, if $\boldsymbol{\theta} = \{\theta_1^k, \theta_2^k \mid 1 \le k \le K\}$ and λ is a scalar parameter, then one can choose a linear combination $W^k = \theta_1^k + \lambda\theta_2^k$. Moreover, AGGR is determined by the shift S_λ associated with G_λ.

In general, as λ follows an unknown distribution μ, it is hard to find the optimal $\boldsymbol{\theta}$ by minimizing $\mathbb{E}_{\lambda\sim\mu}[L_{\mathbf{X}}(\lambda, \boldsymbol{\theta})]$ directly. On the other hand, the EM algorithm (Bishop, 2006) enables the joint estimation of μ and $\boldsymbol{\theta}$ if we can reformulate the objective as an MLE.

To minimize the loss given $\mathbf{X}$, the parameter $\boldsymbol{\theta}$ is determined by λ and vice versa. Therefore, $\Psi(\cdot, \mathbf{x}; \cdot)$ becomes a random GNN model that depends on $\lambda, \boldsymbol{\theta}$ and input $\mathbf{x}$. We aim to *identify a realization of the random models that makes the observation* $\mathbf{X}$ *likely*, i.e., there is less discrepancy between the estimator labels $\hat{\mathbf{y}}$ and ground truth labels $\mathbf{y}$ measured by the loss $\ell(\hat{\mathbf{y}}, \mathbf{y})$. Motivated by the discussions above, we consider the likelihood function $p(\lambda, \mathbf{X} \mid \boldsymbol{\theta})$ on $\boldsymbol{\theta}$ and formulate the following MLE as our objective:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} p(\mathbf{X} \mid \boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\lambda\sim\mu}[p(\lambda, \mathbf{X} \mid \boldsymbol{\theta})]. \tag{2}$$

Before discussing the main algorithm in subsequent subsections, we preview the roles of μ and $L_{\mathbf{X}}(\cdot,\cdot)$ in the algorithm.

We shall see that the EM algorithm outputs a distribution $\hat{\mu}$ of λ, serving as an estimate of μ, by leveraging the PAC-Bayesian framework (Guedj, 2019). In this framework, the density of λ is proportional to the Gibbs posterior, depending on $\ell(\cdot,\cdot)$. Consequently, $\hat{\mu}$ assigns higher probability density to λ when the loss $L_{\mathbf{X}}(\lambda,\boldsymbol{\theta}^*)$ is lower. Therefore, we are minimizing the given loss as a main component of the algorithm. In the above formulation, $\boldsymbol{\theta}$ is the parameter and μ is unknown before attaining $\boldsymbol{\theta}$. However, once $\boldsymbol{\theta}$ is determined, an estimation of μ is obtained using the Gibbs posterior.

Example 1. *Assume that μ is the delta distribution δ_{λ_0} supported on λ_0, so that the graph G_{λ_0} is deterministic. If we consider the Gibbs posterior, then we have $p(\mathbf{X}\mid\boldsymbol{\theta}) \propto \exp(-\eta\ell(\hat{\mathbf{y}},\mathbf{y}))$ for a hyperparameter η, where $\hat{\mathbf{y}}$ depends on both $\mathbf{X}=\{\mathbf{x},\mathbf{y}\}$ and $\boldsymbol{\theta}$. Thus, maximizing $p(\mathbf{X}\mid\boldsymbol{\theta})$ is equivalent to the classical objective of minimizing $\ell(\hat{\mathbf{y}},\mathbf{y})$.*

4. The Proposed Method and Its Derivation

4.1. EM for GNN

Optimizing (2) directly can be challenging, and we utilize the EM algorithm that employs an iterative approach alternating between the E-step and the M-step. Adapted to our setting, the process unfolds as follows:

(a) E-step: Given parameters $\boldsymbol{\theta}^{(t)}$ at the t-th iteration, we compute the expectation as the Q-function:

$$Q(\boldsymbol{\theta}\mid\boldsymbol{\theta}^{(t)}) = \mathbb{E}_{\lambda\sim p(\cdot\mid\mathbf{X},\boldsymbol{\theta}^{(t)})}[\log p(\lambda,\mathbf{X}\mid\boldsymbol{\theta})]. \quad (3)$$

(b) M-step: $\boldsymbol{\theta}^{(t+1)}$ is updated as $\arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}\mid\boldsymbol{\theta}^{(t)})$.

The E-step: In the t-th iteration, in the same spirit as the PAC-Bayesian framework (Guedj, 2019), we apply the Gibbs posterior and assume that

$$p(\lambda,\mathbf{X}\mid\boldsymbol{\theta}) \propto \exp(-\eta^{(t)}L_{\mathbf{X}}(\lambda,\boldsymbol{\theta}))\pi_0(\lambda,\mathbf{X}), \quad (4)$$

for a tunable hyperparameter $\eta^{(t)}$, while $\pi_0(\cdot)$ is a prior density of the joint $(\lambda,\mathbf{X})$ independent of $\boldsymbol{\theta}$, representing our initial knowledge regarding λ and $\mathbf{X}$. In this expression, $L_{\mathbf{X}}(\lambda,\boldsymbol{\theta})$ implicitly depends on the observations $\mathbf{X}$. The normalization constant is given by

$$\begin{aligned} C(\boldsymbol{\theta}) &= \int_{(\lambda,\mathbf{X}')\in\Lambda\times\mathfrak{X}} \exp(-\eta^{(t)}L_{\mathbf{X}'}(\lambda,\boldsymbol{\theta}))\pi_0(\lambda,\mathbf{X}')\,\mathrm{d}(\lambda,\mathbf{X}') \\ &= \mathbb{E}_{(\lambda,\mathbf{X}')\sim\pi_0}\Big[\exp(-\eta^{(t)}L_{\mathbf{X}'}(\lambda,\boldsymbol{\theta}))\Big]. \end{aligned} \quad (5)$$

As a prior belief, we treat the observed $\mathbf{X}$ as a typical sample such that the above average is (approximately) the same as the average over graphs by fixing $\mathbf{X}$ (cf. Appendix H.1). We assume that for each fixed $\mathbf{X}$, there exists some prior distribution with density $p_{0,\mathbf{X}}(\cdot)$ on Λ such that:

$$\begin{aligned} &\mathbb{E}_{(\lambda,\mathbf{X}')\sim\pi_0}\Big[\exp(-\eta^{(t)}L_{\mathbf{X}'}(\lambda,\boldsymbol{\theta}))\Big] \\ \approx &\int_{\lambda\in\Lambda} \exp(-\eta^{(t)}L_{\mathbf{X}}(\lambda,\boldsymbol{\theta}))p_{0,\mathbf{X}}(\lambda)\,\mathrm{d}(\lambda) \\ = &\mathbb{E}_{\lambda\sim p_{0,\mathbf{X}}}\Big[\exp(-\eta^{(t)}L_{\mathbf{X}}(\lambda,\boldsymbol{\theta}))\,\Big|\,\mathbf{X}\Big]. \end{aligned} \quad (6)$$

For notational simplicity, we use $p_0(\cdot)$ to denote $p_{0,\mathbf{X}}(\cdot)$. Correspondingly, $\mathbb{E}_{\lambda\sim p_0}\big[\exp(-\eta^{(t)}L_{\mathbf{X}}(\lambda,\boldsymbol{\theta}))\,\big|\,\mathbf{X}\big]$ is denoted by $\mathbb{E}_{\lambda\sim p_0}\big[\exp(-\eta^{(t)}L_{\mathbf{X}}(\lambda,\boldsymbol{\theta}))\big]$. Hence, we have

$$C(\boldsymbol{\theta}) = \mathbb{E}_{\lambda\sim p_0}\Big[\exp(-\eta^{(t)}L_{\mathbf{X}}(\lambda,\boldsymbol{\theta}))\Big]. \quad (7)$$

On the other hand, given $\boldsymbol{\theta}^{(t)}$, we estimate $p(\lambda\mid\mathbf{X},\boldsymbol{\theta}^{(t)})$ in the subscript of $\mathbb{E}$ in (3). From (4), we have

$$\begin{aligned} p(\lambda\mid\mathbf{X},\boldsymbol{\theta}^{(t)}) &= \frac{p(\lambda,\mathbf{X}\mid\boldsymbol{\theta}^{(t)})}{p(\mathbf{X}\mid\boldsymbol{\theta}^{(t)})} \\ &\propto \exp(-\eta^{(t)}L_{\mathbf{X}}(\lambda,\boldsymbol{\theta}^{(t)}))\frac{\pi_0(\lambda,\mathbf{X})}{p(\mathbf{X}\mid\boldsymbol{\theta}^{(t)})}. \end{aligned}$$

We assume that there is a prior $p'_{0,t}(\cdot)$ such that $p'_{0,t}(\lambda) \propto \frac{\pi_0(\lambda,\mathbf{X})}{p(\mathbf{X}\mid\boldsymbol{\theta}^{(t)})}$, which is independent of $\boldsymbol{\theta}$. However, it is a function of t as $\boldsymbol{\theta}^{(t)}$ depends on t. By fixing $\mathbf{X}$, the posterior is written as

$$p(\lambda\mid\mathbf{X},\boldsymbol{\theta}^{(t)}) \propto \exp(-\eta^{(t)}L_{\mathbf{X}}(\lambda,\boldsymbol{\theta}^{(t)}))p'_{0,t}(\lambda). \quad (8)$$

In our framework, we do not need to estimate the normalization constant for (8).

Remark 1. *From the above discussion, we see that priors $p_0(\cdot)$ and $p'_{0,t}(\cdot)$ play important roles. We discuss their choices in Section 5 below. However, it is always desirable to have a weaker prior assumption, under which the optimizer can still be readily estimated.*

The M-step: We now analyze the Q-function in more detail. *For convenience, we use $p_t(\lambda)$ to denote $p(\lambda\mid\mathbf{X},\boldsymbol{\theta}^{(t)})$.*

Combining (7) and (8), we express $Q(\boldsymbol{\theta}\mid\boldsymbol{\theta}^{(t)})$ in (3) as:

$$\begin{aligned} &Q(\boldsymbol{\theta}\mid\boldsymbol{\theta}^{(t)}) \\ =&\mathbb{E}_{\lambda\sim p_t}\left[\log\frac{\exp(-\eta^{(t)}L_{\mathbf{X}}(\lambda,\boldsymbol{\theta}))\pi_0(\lambda,\mathbf{X})}{C(\boldsymbol{\theta})}\right] \\ =&-\eta^{(t)}\mathbb{E}_{\lambda\sim p_t}[L_{\mathbf{X}}(\lambda,\boldsymbol{\theta})] + D - \log C(\boldsymbol{\theta}), \end{aligned}$$

where D is a constant independent of $\boldsymbol{\theta}$.

To estimate $\log C(\boldsymbol{\theta})$, consider the Jensen inequality:

$$-\eta^{(t)}\mathbb{E}_{\lambda\sim p_0}[L_{\mathbf{X}}(\lambda,\boldsymbol{\theta})] \le \log C(\boldsymbol{\theta}).$$

This means that if $\log C(\boldsymbol{\theta})$ is small, then necessarily so is $-\eta^{(t)}\mathbb{E}_{\lambda\sim p_0}[L_{\mathbf{X}}(\lambda,\boldsymbol{\theta})]$. On the other hand, (Teh et al., 2006) proposes to use $\mathbb{E}[\log Y] + \dfrac{\operatorname{var}(Y)}{2\mathbb{E}(Y)^2}$ to approximate $\log\mathbb{E}[Y]$ for a random variable Y. This is derived from the second-order Taylor expansion of $\log Y$ at $\log\mathbb{E}[Y]$. In our case, we have

$$\begin{aligned}\log C(\boldsymbol{\theta}) &\approx -\eta^{(t)}\mathbb{E}_{\lambda\sim p_0}[L_{\mathbf{X}}(\lambda,\boldsymbol{\theta})] \\ &+\frac{\operatorname{var}\big(\exp(-\eta^{(t)}L_{\mathbf{X}}(\lambda,\boldsymbol{\theta}))\big)}{2\big(\mathbb{E}_{\lambda\sim p_0}\big[\exp(-\eta^{(t)}L_{\mathbf{X}}(\lambda,\boldsymbol{\theta}))\big]\big)^2}.\end{aligned} \tag{9}$$

If $-\eta^{(t)}\mathbb{E}_{\lambda\sim p_0}[L_{\mathbf{X}}(\lambda,\boldsymbol{\theta})]$ is the dominant component, then we may use $-\eta^{(t)}\mathbb{E}_{\lambda\sim p_0}[L_{\mathbf{X}}(\lambda,\boldsymbol{\theta})]$ as a proxy for $\log C(\boldsymbol{\theta})$, which is more manageable. In Appendix H.2, we numerically verify that this is indeed the case for our applications.

Hence, $Q(\boldsymbol{\theta}\mid\boldsymbol{\theta}^{(t)})$ is approximated by

$$-\eta^{(t)}\Big(\mathbb{E}_{\lambda\sim p_t}[L_{\mathbf{X}}(\lambda,\boldsymbol{\theta})]-\mathbb{E}_{\lambda\sim p_0}[L_{\mathbf{X}}(\lambda,\boldsymbol{\theta})]\Big)+D.$$

In summary, if we disregard $\eta^{(t)}$ and D, which are independent of $\boldsymbol{\theta}$, we may minimize the following in the M-step:

$$\begin{aligned}J(\boldsymbol{\theta}) &= \mathbb{E}_{\lambda\sim p_t}[L_{\mathbf{X}}(\lambda,\boldsymbol{\theta})]-\mathbb{E}_{\lambda\sim p_0}[L_{\mathbf{X}}(\lambda,\boldsymbol{\theta})] \\ &= \int_{\lambda\in\Lambda}\big(p_t(\lambda)-p_0(\lambda)\big)L_{\mathbf{X}}(\lambda,\boldsymbol{\theta})\,\mathrm{d}\lambda.\end{aligned} \tag{10}$$

4.2. The Proposed Algorithm: EMGNN

To minimize $J(\boldsymbol{\theta})$ in (10), our strategy is to re-express it as an expectation. For this purpose, we introduce a proposal distribution. Let $q(\cdot)$ be the density function of a probability distribution on the sample space Λ whose support includes that of p_0. Then we have:

$$\begin{aligned}J(\boldsymbol{\theta}) &= \int_{\lambda\in\Lambda} q(\lambda)\frac{p_t(\lambda)-p_0(\lambda)}{q(\lambda)}L_{\mathbf{X}}(\lambda,\boldsymbol{\theta})\,\mathrm{d}\lambda \\ &= \mathbb{E}_{\lambda\sim q}\left[\frac{p_t(\lambda)-p_0(\lambda)}{q(\lambda)}L_{\mathbf{X}}(\lambda,\boldsymbol{\theta})\right].\end{aligned}$$

We propose to minimize $J(\boldsymbol{\theta})$ by first randomly drawing samples $\Lambda_{T'}=\{\lambda_1,\ldots,\lambda_{T'}\}$ according to the density $q(\cdot)$. Following that, we successively apply gradient descent to $\frac{p_t(\lambda_{t'})-p_0(\lambda_{t'})}{q(\lambda)}L_{\mathbf{X}}(\lambda_{t'},\boldsymbol{\theta})$ to update $\boldsymbol{\theta}$. Finally, given (8), $p_t(\lambda)$ can be approximated by an empirical distribution if we apply an MCMC method. The overall algorithm is summarized in Algorithm 1 and illustrated in Figure 2.

Remark 2. *In practice, the choices of the prior distributions $p_0(\cdot), q(\cdot)$ and $p'_{0,t}(\cdot)$ are hyperparameters. Moreover, in our experiments, $p'_{0,t}(\cdot)$ is set to be the same for every t. We also discretize the continuous sample space Λ for simplicity in analysis and computation.*

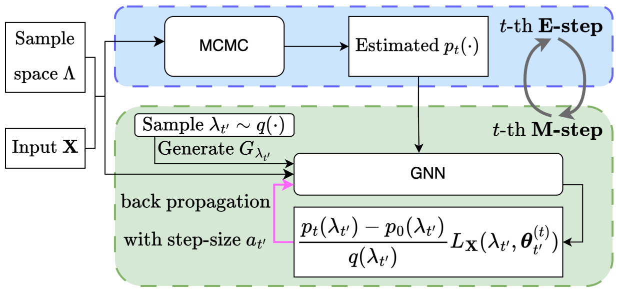

Sample space Λ
Input $\mathbf{X}$
MCMC
Estimated $p_t(\cdot)$
t-th **E-step**
t-th **M-step**
Sample $\lambda_{t'}\sim q(\cdot)$
Generate $G_{\lambda_{t'}}$
GNN
back propagation
with step-size $a_{t'}$
$\frac{p_t(\lambda_{t'})-p_0(\lambda_{t'})}{q(\lambda_{t'})}L_{\mathbf{X}}(\lambda_{t'},\boldsymbol{\theta}_{t'}^{(t)})$

Figure 2. Illustration of EMGNN.

Algorithm 1 EMGNN

Input: The observed graph G,
The node features $\mathbf{x}$,
The number of EM iterations T,
The number of epochs per M-step T',
The sample space Λ,
Prior distributions $p_0, p'_{0,t}$ and q,
Function to convert samples to empirical distribution $g(\cdot)$,
Task-specific function to generate λ influenced graphs $h(\cdot)$,
A non-increasing step-size $a_{t'}$.
Output: The learned representation $\mathbf{z}$.
Initialization: Warm up Ψ using G.

```
1:  for $t = 1$ to $T$ do
2:    AcceptedList$^{(t)} \leftarrow$ MCMC($\Lambda$, $p'_{0,t}$)
3:    EmpProbDict$^{(t)} \leftarrow g$(AcceptedList$^{(t)}$)
4:    for $t' = 1$ to $T'$ do
5:      Sample $\lambda_{t'} \sim q$.
6:      $G_{\lambda_{t'}} \leftarrow h(\lambda_{t'}, G)$
7:      Update via gradient descent: $\boldsymbol{\theta}_{t'+1}^{(t)} = \boldsymbol{\theta}_{t'}^{(t)} - a_{t'}\nabla(\frac{p_t(\lambda_{t'})-p_0(\lambda_{t'})}{q(\lambda_{t'})}L_{\mathbf{X}}(\lambda_{t'},\boldsymbol{\theta}_{t'}^{(t)}))$
8:      $\boldsymbol{\theta}_{t'+1}^{(t)}, \mathbf{z}_{t'} \leftarrow \Psi(\lambda_{t'}, \mathbf{x}; \boldsymbol{\theta}_{t'}^{(t)})$
9:    end for
10:   $\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}_{T'}^{(t)}$
11:   $t \leftarrow t+1$
12: end for
13: $\mathbf{z}_{\text{final}} = \mathbb{E}_{\lambda\sim p_T(\cdot)}\Big[\Psi(\lambda, \mathbf{x}; \boldsymbol{\theta}_{T'}^{(T)})\Big]$
```

Remark 3. *If we algorithmically plug in the delta distribution supported on λ_0 and $p_0(\lambda_0)=0$ for $p'_{0,t}(\cdot)$ and $q(\cdot)$ respectively, then EMGNN reduces to the ordinary GNN model on the graph G_{λ_0}.*

Remark 4. *Note that for the coefficient $\frac{p_t(\lambda)-p_0(\lambda)}{q(\lambda)}$, if $p_t(\lambda)<p_0(\lambda)$, then the loss $L_{\mathbf{X}}(\lambda,\boldsymbol{\theta})$ is to be made larger. Intuitively, in this case, a "bad" λ is chosen. For the choice of $q(\cdot)$, in practice, we propose two options in Section 5: either the uniform distribution or $q(\cdot)=p_t(\cdot)$. Nonetheless, $q(\cdot)$ can also be other appropriate density functions.*

As we do not minimize $J(\boldsymbol{\theta})$ directly, we justify the proposed approach under additional assumptions. We theoretically analyze the performance of the proposed (randomized) algorithm in lines 5-12 of Algorithm 1, denoted by $\mathcal{A}$. With samples $\Lambda_{T'}$, the algorithm $\mathcal{A}$ outputs $\widehat{\boldsymbol{\theta}} = \mathcal{A}(\Lambda_{T'})$. The following expression is considered in algorithm $\mathcal{A}$:

$$J_{\Lambda_{T'}}(\widehat{\boldsymbol{\theta}}) = \frac{1}{T'} \sum_{\lambda_{t'} \in \Lambda_{T'}} \frac{p_t(\lambda_{t'}) - p_0(\lambda_{t'})}{q(\lambda_{t'})} L_{\mathbf{X}}(\lambda_{t'}, \widehat{\boldsymbol{\theta}}).$$

We assume that after translation and scaling by positive constant of $L_{\mathbf{X}}(\lambda, \cdot)$ if necessary, the expression $\frac{p_t(\lambda)-p_0(\lambda)}{q(\lambda)} L_{\mathbf{X}}(\lambda, \boldsymbol{\theta})$ always belong to $[0,1]$. The following notions are well-known.

Definition 1. *A differentiable function f is α-Lipschitz if for all x in the domain of f, we have $\|\nabla f(x)\| \leq \alpha$. It is β-smooth if its gradient is β-Lipschitz.*

Denote by $\mathbf{1}_{\{p_t(\cdot) \geq p_0(\cdot)\}}(\lambda)$ the indicator that is 1 if $p_t(\lambda) \geq p_0(\lambda)$, and 0 otherwise. Let $b_1 = \mathbb{E}_{\lambda \sim q}\left[\mathbf{1}_{\{p_t(\cdot) \geq p_0(\cdot)\}}\right]$. Intuitively, it computes the measure of λ, for which $p_t(\cdot)$ is larger. On the other hand, let $b_2 = \mathbb{E}_{\lambda \sim q}\left[\frac{|p_t(\lambda) - p_0(\lambda)|}{q(\lambda)}\right]$, and $\gamma = \sup_{\lambda \in \Lambda} 1/q(\lambda)$.

Theorem 2. *Assume for any λ, the loss $L_{\mathbf{X}}(\lambda, \cdot)$ is convex, α-Lipschitz and β-smooth. Let b_1, b_2, γ be defined as above. If for every $t' \leq T'$, the non-increasing step-size in the algorithm $\mathcal{A}$ satisfies $a_{t'} \leq \min\{2/(\beta\gamma), c/t'\}$ for a constant c, then there is a constant C independent of T', α such that*

$$\left|\mathbb{E}_{\mathcal{A}, \Lambda_{T'}}\left[J_{\Lambda_{T'}}(\widehat{\boldsymbol{\theta}}) - J(\widehat{\boldsymbol{\theta}})\right]\right| \leq \epsilon = C\left(\frac{b_2^2 \alpha^2}{T'}\right)^{\frac{1}{\beta\gamma c(1-b_1)+1}}.$$

Remark 5. *From the result, we see that if b_1 is close to 1, i.e, the set $\{\lambda \mid p_t(\lambda) \geq p_0(\lambda)\}$ has a large measure, then the expected error decays at a rate close to T'^{-1}.*

4.3. A Brief Discussion on Testing

As our framework deals with a distribution of graphs, during testing, the final learned representation is $\mathbf{z}_{\text{final}} = \mathbb{E}_{\lambda \sim p_T}\left[\Psi(\lambda, \mathbf{x}; \boldsymbol{\theta}_{T'}^{(T)})\right]$ (cf. Theorem 1). The learned model parameters are a particular realization of the possible random models that align with the observed data $\mathbf{X}$ and the multiple graphs influence the final embeddings based on their respective likelihoods. The embedding $\mathbf{z}_{\text{final}}$ is subjected to a softmax operation to obtain $\hat{\mathbf{y}}$ for node classification tasks, while a READOUT function is applied for graph-level tasks.

5. EXPERIMENTS

We study node classification for heterogeneous and homogeneous graphs. In Appendix B and Appendix E, we explore chemical datasets and out-of-distribution (OOD) detection, respectively. Additionally, in Appendix C and Appendix D, we provide dataset and implementation details, and discuss model complexity.

5.1. Heterogeneous Graphs

5.1.1. The Experimental Setup and Baselines

As outlined in Section 3.1, it is more natural to apply the framework to heterogeneous graphs. Let G be such a graph and assume with ω edge types. To apply the framework, it suffices to specify the sample parameter space. We introduce a vector of latent parameters $\boldsymbol{\lambda} = \{\lambda_1, \ldots, \lambda_\omega\}$, where each $\lambda_i \in [0,1]$ and $\sum_{i=1}^{\omega} \lambda_i = 1$. The number λ_i is the weight for the i-th edge type. Hence, the sample parameter space is the $(w-1)$-simplex, denoted by Λ. For a chosen $\boldsymbol{\lambda}$, the associated weighted graph $G_{\boldsymbol{\lambda}}$ has adjacency matrix

$$A_{\boldsymbol{\lambda}} = \sum_{i=1}^{\omega} \lambda_i A_i, \tag{11}$$

where A_i is the edge type specific adjacency matrix corresponding to the i-th edge type. Note that when any $\lambda_i = 0$, the edges of the associated edge types are removed. For our model, we discretize the interval $[0,1]$ in increments of 0.05, and the resulting discretized sample parameter space is denoted by $\widehat{\Lambda}$.

The heterogeneous graph datasets used are the same as those in Yun et al. (2019) and Lee et al. (2022). These datasets include two citation networks DBLP and ACM, as well as a movie dataset IMDB. They have similar edge type structures (cf. Appendix C.1). For example, IMDB has three node types (Movie (M), Actor (A), Director (D)) and two edge types (MD, MA). Hence, the parameter is $\boldsymbol{\lambda} = (\lambda, 1-\lambda)$.

We assess our approach against seven baseline models. Specifically, GAT, GCN, LSM_GCN (Ma et al., 2019) and SBM_GCN (Ma et al., 2019) are designed for homogeneous graphs, while GTN (Yun et al., 2019), Simple-HGN (Lv et al., 2021) and SeHGNN (Yang et al., 2023) are state-of-the-art models developed for heterogeneous graphs. Our framework, applicable to diverse GNNs, is exemplified with a GCN backbone, named EM-GCN. We consider different variants of EM-GCN, based on choices of $p_0(\cdot), p'_{0,t}(\cdot), q(\cdot)$, as summarized in Table 1.

Table 1. Variants of EM-GCN with different $p_0(\cdot), p'_{0,t}(\cdot), q(\cdot)$. For the nomenclature, we use 'u' for the uniform distribution, 'd' for the delta distribution, and 't' for the adaptive distribution that depends on t.

$p_0(\cdot)$	$p'_{0,t}(\cdot)$	$q(\cdot)$	Model
Unif($\widehat{\Lambda}$)	Unif($\widehat{\Lambda}$)	$p_t(\cdot)$	EM-GCN[uut]
Unif($\widehat{\Lambda}$)	Unif($\widehat{\Lambda}$)	Unif($\widehat{\Lambda}$)	EM-GCN[uuu]
δ_{λ_0}	Unif($\widehat{\Lambda}$)	$p_t(\cdot)$	EM-GCN[dut]
δ_{λ_0}	Unif($\widehat{\Lambda}$)	Unif($\widehat{\Lambda}$)	EM-GCN[duu]

Table 2. Heterogeneous node classification task. The results shown are averaged over ten runs and accompanied by the standard deviation. The best performance is boldfaced and the second-best performance is underlined.

	IMDB		ACM		DBLP	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
GCN	61.91 ± 0.67	60.91 ± 0.57	91.92 ± 0.40	92.00 ± 0.41	94.60 ± 0.31	93.88 ± 0.36
GAT	63.54 ± 1.10	61.87 ± 0.95	92.61 ± 0.36	92.68 ± 0.36	94.48 ± 0.22	93.74 ± 0.27
LSM_GCN	62.65 ± 1.73	61.84 ± 1.55	92.03 ± 0.23	92.09 ± 0.24	89.38 ± 0.30	88.35 ± 0.32
SBM_GCN	63.43 ± 0.60	62.50 ± 0.43	91.85 ± 0.33	91.93 ± 0.34	89.40 ± 0.33	88.37 ± 0.34
GTN	60.58 ± 2.10	59.12 ± 1.58	92.12 ± 0.62	92.23 ± 0.60	94.17 ± 0.26	93.59 ± 0.40
Simple-HGN	58.91 ± 1.06	58.30 ± 0.34	**92.73 ± 0.21**	92.56 ± 0.42	94.48 ± 0.38	93.69 ± 0.32
SeHGNN	62.13 ± 2.38	60.62 ± 1.95	92.45 ± 0.17	92.51 ± 0.16	94.86 ± 0.14	94.14 ± 0.19
EM-GCN[uut]	**64.78 ± 1.24**	**63.36 ± 0.80**	92.70 ± 0.26	**92.78 ± 0.26**	**95.06 ± 0.39**	**94.41 ± 0.45**
EM-GCN[uuu]	63.35 ± 0.79	62.25 ± 0.59	92.35 ± 0.38	92.45 ± 0.38	94.95 ± 0.24	94.28 ± 0.28
EM-GCN[dut]	62.49 ± 0.87	61.55 ± 0.71	92.31 ± 0.43	92.41 ± 0.42	94.89 ± 0.17	94.15 ± 0.23
EM-GCN[duu]	62.01 ± 0.55	61.15 ± 0.46	92.18 ± 0.52	92.29 ± 0.52	95.02 ± 0.19	94.34 ± 0.20

5.1.2. Results

Results are shown in Table 2. Similar to recent findings (Lv et al., 2021), GCN and GAT are observed to perform competitively against models designed for heterogeneous graphs such as GTN under appropriate settings. Meanwhile, EM-GCN[uut] consistently outperforms other variants in our framework, in both micro and macro F1 scores. In particular, the superior performance of EM-GCN[uut], EM-GCN[uuu], EM-GCN[dut], and EM-GCN[duu] compared to GCN indicates the effectiveness of our distribution-based framework.

EM-GCN[uut] also often surpasses baselines models with attention mechanisms, namely GAT, Simple-HGN, SeHGNN, and GTN, despite not incorporating any attention mechanisms. This could be attributed to the construction of multiple graphs, which may form instances whose information is similar to what is achieved with semantic attention. In addition, the model may extract additional useful interactions from other graph instances, enhancing its performance.

Both LSM_GCN and SBM_GCN are based on the variational approach. These two methods employ explicit parametric graph distribution models, namely, latent space models (LSM) and stochastic block models (SBM). The training procedure estimates the (deterministic) parameters of the distribution model, which then determines the graph distribution. In principle, this is different from our framework, where we parameterize the graphs and learn a distribution on the sample space of parameters, which then determines a distribution of graphs. Our approach can be more flexible as no explicit graph distribution model is assumed. As expected, we observe that EM-GCN[uut] consistently outperforms LSM_GCN and SBM_GCN. However, LSM_GCN and SBM_GCN are computationally more efficient.

5.1.3. Further Analysis

Ablation study: For EM-GCN[dut] and EM-GCN[duu], λ_0 for the delta function is set to be any $\lambda \in \Lambda \backslash \widehat{\Lambda}$. Consequently, $p_0(\cdot)$ will be 0 with probability 1 w.r.t $q(\cdot)$ on $\widehat{\Lambda}$. Hence, for these variants, there is no "bad" λ such that the corresponding iteration increases $L_{\mathbf{X}}(\cdot, \cdot)$ (cf. Remark 4).

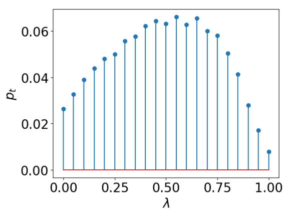
From Table 2, we see that EM-GCN[uut] outperforms EM-GCN[dut], along with EM-GCN[uuu] frequently outperforming EM-GCN[duu]. They indicate that increasing the loss for a "bad" λ is beneficial as it penalizes deviations from desirable graphs.

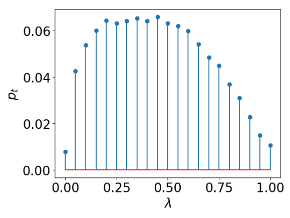
The learned distribution: We examine the learned empirical distributions, depicted in Figure 3. Across all datasets, we notice that the empirical probability of λ is relatively high within the range of approximately $[0.4, 0.6]$. This suggests a possible explanation for the decent performance of GCN on a single graph with uniform edge weights.

For IMDB, (11) is of the form $\lambda A_{\text{MD}} + (1-\lambda) A_{\text{MA}}$. We observe that $\lambda = 1$ has a relatively lower probability compared to $\lambda = 0$. When $\lambda = 1$, it implies that edges in A_{MA} are all removed. This indicates that the MA relation is more crucial than the MD relation. This observation might be due to MA having an edge density more than triple that of MD. Similarly, for the ACM dataset, where the disparity in edge density is also substantial, (11) takes the form $\lambda A_{\text{PA}} + (1-\lambda) A_{\text{PS}}$. Here, P, A, S stand for Paper, Author, Subject. The shift of λ towards 1 indicates that the PA relation is more significant than PS, and agrees with the higher density of PA type. The results demonstrate that our approach implicitly captures such key graph features.

Table 3. Node classification on homogeneous graphs following the setup of Zhang et al. (2019).

	Cora			Citeseer			Pubmed		
	5 labels	10 labels	20 labels	5 labels	10 labels	20 labels	5 labels	10 labels	20 labels
GCN	74.62 ± 0.54	75.30 ± 0.47	81.37 ± 0.31	54.24 ± 1.26	66.07 ± 0.68	70.19 ± 0.46	69.96 ± 0.65	72.96 ± 0.58	78.45 ± 0.44
DropEdge	74.79 ± 0.56	75.88 ± 0.19	81.46 ± 0.64	54.44 ± 2.54	67.59 ± 1.41	71.23 ± 1.26	71.69 ± 0.50	73.14 ± 0.33	78.50 ± 0.54
RSGNN	**76.80 ± 3.19**	**78.23 ± 2.62**	80.79 ± 0.91	**59.97 ± 1.89**	68.41 ± 0.94	69.73 ± 0.53	70.45 ± 0.78	70.92 ± 0.86	77.55 ± 0.46
BGCN	75.97 ± 0.54	76.52 ± 0.50	81.18 ± 0.48	56.58 ± 0.96	70.61 ± 0.69	72.11 ± 0.40	70.51 ± 1.61	73.36 ± 1.23	76.55 ± 0.65
EM-GCN	74.44 ± 0.76	76.71 ± 0.46	**82.24 ± 0.48**	58.04 ± 2.24	**70.65 ± 1.10**	**72.13 ± 0.96**	**74.03 ± 0.55**	**74.93 ± 0.24**	**78.96 ± 0.38**


(a) IMDB


(b) DBLP

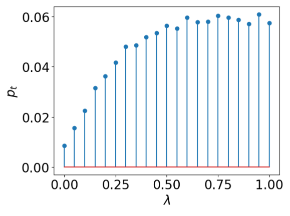

(c) ACM

Figure 3. Empirical distribution of $p_t(\cdot)$ from the final E-step.

5.2. Homogeneous Graphs

5.2.1. The Experimental Setup and Baslines

For $G = (V, E)$, we follow Section 3.1 for the design of Λ. Specifically, we propose to parametrize graphs by a pair $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2\}$ with $\lambda_1, \lambda_2 \in [0, 0.2]$. Here, λ_1 is the probability of randomly removing edges from the graph, to account for possible "noisy edges" in the observed graphs. On the other hand, λ_2 is the probability of randomly introducing edges from a pre-constructed subset E' of all missing edges. More specifically, there are $n(n-1)/2$ possible edges between pairs of distinct nodes of size $|V| = n$. For each pair of nodes v, v' without an edge connection in G, we compute the cosine similarity of their node features. The edge set E' is obtained from including node pairs whose cosine similarities are above a threshold τ (see Appendix C.2). It is intuitively considered as the set of "likely" missing edges based on feature similarities. Notice that $\boldsymbol{\lambda}$ does not determine a unique graph, we need to slightly modify Algorithm 1 when applying MCMC (see Appendix C.4).

The most relevant benchmark is BGCN (Zhang et al., 2019) for homogeneous graphs, based on a Bayesian approach to infer the graph distribution. As our construction of Λ involves edge removal and addition, we also consider DropEdge (Rong et al., 2020) and RSGNN (Dai et al., 2022), where the former randomly removes edges at each epoch and the latter learns a denser graph using a trained link predictor. The experimental setup follows exactly that from Zhang et al. (2019), wherein for each dataset, we evaluate the performance of the algorithms under limited data scenarios where only 10 or 5 labels per class are available.

5.2.2. Results

Based on Table 2 and the ablation study in Section 5.1.3, we use the EM-GCN[uut] variant for EM-GCN. Results are shown in Table 3. Overall, across all datasets, EM-GCN surpasses the performance of BGCN and DropEdge and frequently outperforms RSGNN. In principle, DropEdge and RSGNN do not leverage the potential of a distribution of graphs, which is fundamentally different from our approach.

5.2.3. Further Analysis

Heterophilic graphs: Recall that a graph is heterophilic if many edges are connecting nodes with different labels. In particular, nodes from the same class are not grouped. Hence, as BGCN has a clustering mechanism, we expect that it might face challenges for heterophilic graphs. On the other hand, based on the construction of Λ, EM-GCN may generate $\boldsymbol{\lambda}$ such that the associated graph reduces inter-class edges while adding intra-class edges. Hence, we expect EM-GCN should significantly outperform BGCN for heterophilic graphs, which is verified by results in Table 4.

EM-GCN is based on the "unsuitable" backbone GCN, which suffers from the same problem as BGCN. Even so, the performance of EM-GCN is much closer to benchmarks ACM-GCN+ (Luan et al., 2022), ACMP (Wang et al., 2023) and GBK-GNN (Du et al., 2022) dedicated to heterophilic graphs. This validates our proposed construction of Λ. Furthermore, our framework can be applied to other backbone models, potentially leading to performance improvements. Therefore, we introduce a variant of our model, EM-ACM, with ACM-GCN+ as the backbone model. From Table 4, we see that EM-ACM generally outperforms its SOTA backbone. This can be reasonably attributed to the use of a distribution of graphs.

Meanwhile, GBK-GNN incorporates attention mechanisms

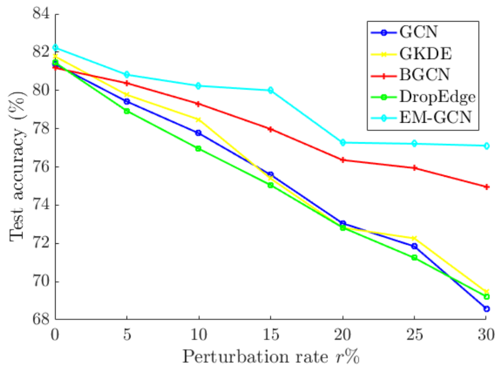


Figure 4. Performance comparison for graph perturbation.

and relies on a bi-kernel to capture both positive and negative adjacent node correlations. While GBK-GNN does outperform its simplified counterpart GAT, EM-ACM demonstrates an overall superior performance. Notably, even EM-GCN outperforms GBK-GNN on the Wisconsin dataset.

Table 4. Node classification on heterophilic graphs. The setup and data splits follow Pei et al. (2020).

	Texas	Wisconsin	Cornell
GAT	52.20 ± 6.60	49.40 ± 4.10	61.90 ± 5.10
GCN	55.14 ± 5.16	51.76 ± 3.06	60.54 ± 5.30
DropEdge	57.57 ± 4.94	57.45 ± 5.47	60.54 ± 5.30
RSGNN	68.38 ± 5.26	68.82 ± 7.25	60.96 ± 6.90
BGCN	57.96 ± 6.77	61.37 ± 4.72	56.48 ± 6.67
GBK-GNN	81.08 ± 4.88	74.27 ± 2.18	84.21 ± 4.33
ACM-GCN+	86.76 ± 4.26	86.86 ± 2.91	84.05 ± 7.88
ACMP	86.20 ± 3.00	86.10 ± 4.00	**85.40 ± 7.00**
EM-GCN	79.46 ± 4.26	83.73 ± 4.34	77.30 ± 4.10
EM-ACM	**88.38 ± 5.14**	**87.06 ± 2.51**	85.14 ± 6.54

Robustness: As discussed in Section 3.1, our approach might be resistant to errors in the observed graph, as we are not focusing on a single graph. To verify, we consider the Cora graph G and randomly perturb $r\%$ of edges (by adding new edges and removing existing edges) for $5 \leq r \leq 30$. We compare EM-GCN against four other methods: GCN, GKDE (Zhao et al., 2020), BGCN, and DropEdge, while observing only the perturbed graph. From the results in Fig. 4, we observe that EM-GCN outperforms the mentioned baselines, and the gap in accuracies widens as r increases. We also observe that BGCN, which has certain common characteristics to our approach such as being able to output graph distributions, outperforms the other benchmarks. This suggests the usefulness of this type of GNNs.

6. Conclusion

In this paper, we explore using a distribution of parametrized graphs for training a GNN in an EM framework. Through a probabilistic framework, we handle the uncertainty in graph structures stemming from various sources. Our approach enables the model to handle multiple graphs where the prediction loss is utilized to estimate the likelihood of the graphs. The performance is proved as we provide it with a wider array of graphs, which it can then sift through to acquire more valuable information or remove noise.

A limitation of our model is that although it can also improve the performance of attention-based GNNs, the extra benefit is less than GNN models without the attention mechanism. For future work, we shall investigate how to synergize our approach with attention-based GNNs more effectively.

Acknowledgements

The first author is supported by Shopee Singapore Private Limited under the Economic Development Board Industrial Postgraduate Programme (EDB IPP). The programme is a collaboration between Shopee and Nanyang Technological University, Singapore. The third author is supported in part by the Nanyang Technological University SPMS Collaborative Research Award 2022, Singapore Ministry of Education Academic Research Fund Tier 2 grants MOE-T2EP20221-0003 and MOE-T2EP20120-0013. The remaining authors are supported in part by the Singapore Ministry of Education Academic Research Fund Tier 2 grant MOE-T2EP20220-0002, and the National Research Foundation, Singapore and Infocomm Media Development Authority under its Future Communications Research and Development Programme.

Impact Statement

This paper introduces a research work aimed at pushing the boundaries of the Machine Learning field. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Bishop, C. *Pattern Recognition and Machine Learning*. Springer, 2006.

Brody, S., Alon, U., and Yahav, E. How attentive are graph attention networks? In *International Conference on Learning Representations*, 2022.

Cen, Y., Hou, Z., Wang, Y., Chen, Q., Luo, Y., Yu, Z., Zhang, H., Yao, X., Zeng, A., Guo, S., Dong, Y., Yang, Y., Zhang, P., Dai, G., Wang, Y., Zhou, C., Yang, H., and Tang, J. Cogdl: A comprehensive library for graph deep learning. In *Proceedings of the ACM Web Conference 2023 (WWW'23)*, 2023.

Chen, M., Wei, Z., Huang, Z., Ding, B., and Li, Y. Simple and deep graph convolutional networks. In *Proceedings* *of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1725–1735. PMLR, 13–18 Jul 2020.

Chen, Y., Yang, M., Zhang, Y., Zhao, M., Meng, Z., Hao, J., and King, I. Modeling scale-free graphs with hyperbolic geometry for knowledge-aware recommendation. In *Proceedings of the 15th ACM International Conference on Web Search and Data Mining*, WSDM '22, pp. 94–102, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450391320. doi: 10.1145/3488560.3498419. URL https://doi.org/10.1145/3488560.3498419.

Dai, E., Jin, W., Liu, H., and Wang, S. Towards robust graph neural networks for noisy graphs with sparse labels. In *Proc. of the 15th ACM International WSDM Conference*, 2022.

Dong, M. and Kluger, Y. Towards understanding and reducing graph structural noise for gnns. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Du, L., Shi, X., Fu, Q., Ma, X., Liu, H., Han, S., and Zhang, D. Gbk-gnn: Gated bi-kernel graph neural networks for modeling both homophily and heterophily. In *Proceedings of the ACM Web Conference 2022*, WWW '22, pp. 1550–1558, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450390965. doi: 10.1145/3485447.3512201.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, ICML'17, pp. 1263–1272. JMLR.org, 2017a.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1263–1272. PMLR, 06–11 Aug 2017b.

Guedj, B. A primer on PAC-Bayesian learning. *arXiv preprint arXiv:1901.05353*, 2019.

Hardt, M., Recht, B., and Singer, Y. Train faster, generalize better: stability of stochastic gradient descent. In *Proceedings of the 33th International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1225–1234. PMLR, Jun 2016.

Ji, F., Jian, X., and Tay, W. P. On distributional graph signals. *arXiv:2302.11104*, 2023a.

Ji, F., Lee, S., Zhao, K., Tay, W. P., and Yang, J. Distributional signals for node classification in graph neural networks. *arXiv:2304.03507*, 2023b.

Ji, F., Lee, S. H., Meng, H., Zhao, K., Yang, J., and Tay, W. P. Leveraging label non-uniformity for node classification in graph neural networks. In *International Conference on Machine Learning*, 2023c.

Ji, F., Tay, W. P., and Ortega, A. Graph signal processing over a probability space of shift operators. *IEEE Transactions on Signal Processing*, 71:1159–1174, 2023d. doi: 10.1109/TSP.2023.3263675.

Kang, Q., Zhao, K., Song, Y., Wang, S., and Tay, W. P. Node embedding from neural Hamiltonian orbits in graph neural networks. In *Proc. International Conference on Machine Learning*, Haiwaii, USA, Jul. 2023.

Kearnes, S., McCloskey, K., Berndl, M., Pande, V., and Riley, P. Molecular graph convolutions: moving beyond fingerprints. *Journal of Computer-Aided Molecular Design*, pp. 595–608, 2016. doi: 10.1007/s10822-016-9938-8.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations (ICLR)*, 2017.

Lee, S. H., Ji, F., and Tay, W. P. Learning on heterogeneous graphs using high-order relations. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3175–3179, 2021. doi: 10.1109/ICASSP39728.2021.9413417.

Lee, S. H., Ji, F., and Tay, W. P. SGAT: Simplicial graph attention network. In *International Joint Conference on Artificial Intelligence*, 2022.

Li, H., Cao, J., Zhu, J., Liu, Y., Zhu, Q., and Wu, G. Curvature graph neural network. *Information Sciences*, 2021a. doi: 10.1016/j.ins.2021.12.077.

Li, M., Zhou, J., Hu, J., Fan, W., Zhang, Y., Gu, Y., and Karypis, G. Dgl-lifesci: An open-source toolkit for deep learning on graphs in life science. *ACS Omega*, 2021b.

Li, R., Yuan, X., Radfar, M., Marendy, P., Ni, W., O'Brien, T., and Casillas-Espinosa, P. Graph signal processing, graph neural network and graph learning on biological data: A systematic review. *IEEE Reviews in Biomedical Engineering*, PP:1–1, 10 2021c. doi: 10.1109/RBME.2021.3122522.

Liu, W., Zhang, Y., Wang, J., He, Y., Caverlee, J., Chan, P. P. K., Yeung, D. S., and Heng, P.-A. Item relationship graph neural networks for e-commerce. *IEEE Transactions on Neural Networks and Learning Systems*, 33(9): 4785–4799, 2022. doi: 10.1109/TNNLS.2021.3060872.

Luan, S., Hua, C., Lu, Q., Zhu, J., Zhao, M., Zhang, S., Chang, X.-W., and Precup, D. Revisiting heterophily for graph neural networks. *Advances in neural information processing systems*, 35:1362–1375, 2022.

Lv, Q., Ding, M., Liu, Q., Chen, Y., Feng, W., He, S., Zhou, C., Jiang, J., Dong, Y., and Tang, J. Are we really making much progress? revisiting, benchmarking and refining heterogeneous graph neural networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, pp. 1150–1160, 2021. doi: 10.1145/3447548.3467350.

Ma, J., Tang, W., Zhu, J., and Mei, Q. A flexible generative framework for graph-based semi-supervised learning. In *Advances in Neural Information Processing Systems*, pp. 3276–3285, 2019.

Michael Defferrard, Xavier Bresson, P. V. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proc. of the 29th International Conference on Neural Information Processing Systems*, 2016.

Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., and Yang, B. Geom-gcn: Geometric graph convolutional networks, 2020.

Rong, Y., Huang, W., Xu, T., and Huang, J. DropEdge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2020.

Sawhney, R., Agarwal, S., Wadhwa, A., and Shah, R. Exploring the scale-free nature of stock markets: Hyperbolic graph learning for algorithmic trading. In *Proceedings of the Web Conference 2021*, WWW '21, pp. 11–22, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383127. doi: 10.1145/3442381.3450095. URL https://doi.org/10.1145/3442381.3450095.

Shen, C., Luo, J., and Xia, K. Molecular geometric deep learning, 2023.

Shui, Z. and Karypis, G. Heterogeneous molecular graph neural networks for predicting molecule properties. In *2020 IEEE International Conference on Data Mining (ICDM)*, pp. 492–500, Los Alamitos, CA, USA, nov 2020. IEEE Computer Society. doi: 10.1109/ICDM50108.2020.00058. URL https://doi.ieeecomputersociety.org/10.1109/ICDM50108.2020.00058.

Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., and Vandergheynst, P. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013.

Teh, Y. W., Newman, D., and Welling, M. A collapsed variational bayesian inference algorithm for latent dirichlet allocation. In *Proc. of the 19th International Conference on Neural Information Processing Systems*, 2006.

Thomas, N., Smidt, T., Kearnes, S., Yang, L., Li, L., Kohlhoff, K., and Riley, P. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds, 2018.

Topping, J., Giovanni, F. D., Chamberlain, B. P., Dong, X., and Bronstein, M. M. Understanding over-squashing and bottlenecks on graphs via curvature. *International Conference on Learning Representations*, 2022.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph Attention Networks. *International Conference on Learning Representations*, 2018.

Villani, C. *Optimal Transport, Old and New*. Springer, 2009.

Wang, M., Zheng, D., Ye, Z., Gan, Q., Li, M., Song, X., Zhou, J., Ma, C., Yu, L., Gai, Y., Xiao, T., He, T., Karypis, G., Li, J., and Zhang, Z. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019.

Wang, Y., Yi, K., Liu, X., Wang, Y. G., and Jin, S. ACMP: Allen-Cahn message passing with attractive and repulsive forces for graph neural networks. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=4fZc_79Lrqs.

Wu, Q., Chen, Y., Yang, C., and Yan, J. Energy-based out-of-distribution detection for graph neural networks. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=zoz7Ze4STUL.

Xiong, Z., Wang, D., Liu, X., Zhong, F., Wan, X., Li, X., Li, Z., Luo, X., Chen, K., Jiang, H., and Zheng, M. Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism. *Journal of Medicinal Chemistry*, pp. 8749–8760, 2020. doi: 10.1021/acs.jmedchem.9b00959.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

Yang, X., Yan, M., Pan, S., Ye, X., and Fan, D. Simple and efficient heterogeneous graph neural network. In *AAAI Conference on Artificial Intelligence*, 2023.

Ye, Z., Liu, K. S., Ma, T., Gao, J., and Chen, C. Curvature graph network. *Proceedings of the 8th International*

Conference on Learning Representations (ICLR 2020), April 2020.

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., and Leskovec, J. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '18, pp. 974–983, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355520. doi: 10.1145/3219819.3219890. URL https://doi.org/10.1145/3219819.3219890.

Yun, S., Jeong, M., Kim, R., Kang, J., and Kim, H. J. Graph transformer networks. In *Proc. of the 33rd International Conference on Neural Information Processing Systems*, 2019.

Zhang, Y., Pal, S., Coates, M., and Üstebay, D. Bayesian graph convolutional neural networks for semi-supervised classification. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*. AAAI Press, 2019. ISBN 978-1-57735-809-1. doi: 10.1609/aaai.v33i01.33015829. URL https://doi.org/10.1609/aaai.v33i01.33015829.

Zhao, K., Kang, Q., Song, Y., She, R., Wang, S., and Tay, W. P. Graph neural convection-diffusion with heterophily. In *Proc. International Joint Conference on Artificial Intelligence*, Macao, China, Aug 2023.

Zhao, X., Chen, F., Hu, S., and Cho, J.-H. Uncertainty aware semi-supervised learning on graph data, 2020.

This appendix contains details about notations, datasets, implementation, for chemical datasets, out-of-distribution (OOD) detection, model complexity, and mathematical proofs referenced in the main text of the paper.

A. List of notations

For easy reference, we list the most used notations in Table 5.

Table 5. List of notations

Graph, Vertex set, Edge set	G (with subscripts), V, E
Adjacency matrix	A (with subscripts)
Nodes	v, v'
Shift operator	S, S_λ
Sample parameter space	Λ
Sample parameter	λ (with subscripts), $\boldsymbol{\lambda}$
Distributions on the sample space	μ (with subscripts)
Density functions	p, q (with subscripts)
GNN model parameters	$\boldsymbol{\theta}$
Features and labels	$\mathbf{x}, \mathbf{y}, \widehat{\mathbf{y}}, \mathbf{X}$
Loss	$\ell(\cdot,\cdot), L_{\mathbf{X}}(\lambda, \mathbf{x}; \boldsymbol{\theta})$
GNN model	Ψ
Number of GNN layers	K

B. Graph Regression Task on Chemical Datasets

Conventional molecular graph representations mirror a molecule's Lewis structure, with atoms as nodes and chemical bonds as edges. This representation falls short in capturing variations in molecular properties resulting from different three-dimensional (3D) spatial arrangements when molecules share the same topology, as seen in cases like cis-trans isomers. Moreover, molecules inherently possess uncertainty due to their quantum mechanical properties, particularly concerning electron orbitals. Hence, using a distribution of graphs for learning in such cases is a sensible choice.

The process of generating different molecular graphs begins with the acquisition of coarse 3D coordinates of the atoms in a molecule using RDKit[1]. Following that, the interatomic Euclidean distances between all atoms within the molecule are calculated. A parameter, λ is then introduced to define a threshold range, $[0, \lambda]$, for determining node connections and generating multiple graph instances. The notion of employing thresholding based on the interatomic distance between nodes in molecular graphs has been previously documented in works such as (Shui & Karypis, 2020) and (Shen et al., 2023). The former introduced a cut-off distance hyperparameter to construct heterogeneous molecular graphs. Meanwhile, our approach aligns more closely with the latter, where the Vietoris-Rips complex and thresholding are used to form a series of G_λ graphs. However, in (Shen et al., 2023), they utilized five non-overlapping, manually adjusted intervals for thresholding and adopted a computationally intensive multi-channel configuration to learn from the five generated graphs. There are also works such as (Thomas et al., 2018), where molecules are treated as 3D point clouds and a radius is set to specify interacting vertices.

The graph construction process may involve adding new edges, connecting distant nodes, or removing existing edges. Ideally, the model should prioritize "useful" graph realizations and assign a low probability to less beneficial ones, effectively discarding them.

[1]A cheminformatics tool. https://www.rdkit.org

Table 6. Graph regression task on molecular datasets. Average test rmse reported, the lower the better.

	Random split			Scaffold split		
Datasets	FreeSolv	ESOL	Lipophilicity	FreeSolv	ESOL	Lipophilicity
GCN	<u>1.157 ± 0.215</u>	<u>0.652 ± 0.073</u>	0.707 ± 0.030	<u>2.618 ± 0.298</u>	0.876 ± 0.037	0.760 ± 0.009
GAT	1.873 ± 0.522	0.837 ± 0.101	0.704 ± 0.058	2.942 ± 0.591	0.907 ± 0.034	0.777 ± 0.037
Weave	1.497 ± 0.251	0.798 ± 0.088	0.789 ± 0.059	3.129 ± 0.203	1.104 ± 0.063	0.844 ± 0.031
MPNN	1.388 ± 0.404	0.703 ± 0.075	<u>0.640 ± 0.025</u>	2.975 ± 0.775	1.117 ± 0.058	**0.735 ± 0.019**
AttentiveFP	1.275 ± 0.289	0.673 ± 0.085	0.719 ± 0.042	2.698 ± 0.297	<u>0.855 ± 0.029</u>	0.762 ± 0.022
GIN	1.678 ± 0.494	0.792 ± 0.097	0.716 ± 0.073	2.957 ± 0.696	0.990 ± 0.057	0.770 ± 0.021
EM-GCN*	**0.936 ± 0.162**	**0.606 ± 0.041**	**0.639 ± 0.028**	**2.189 ± 0.128**	**0.834 ± 0.027**	<u>0.743 ± 0.013</u>

B.1. Baselines and Datasets

MoleculeNet[2] is a popular benchmark for molecular machine learning, encompassing multiple datasets to be tested on a diverse of molecular properties. For our evaluation, we specifically selected the datasets FreeSolv, ESOL, and Lipophilicity, all of which are designed for graph regression tasks.

We compared our approach against standard models for molecular properties prediction that do not incorporate transfer learning from a larger dataset such as Zinc15[3]. The selected baseline models for this comparison included Weave (Kearnes et al., 2016), MPNN (Gilmer et al., 2017b), AttentiveFP (Xiong et al., 2020), GIN (Xu et al., 2019), as well as the standard GCN and GAT models. For EMGNN, a GNN model that generalizes GCN with a degree-1 convolutional filter $P_\lambda(S_\lambda)$ (refer to Section 2.2) is utilized as the backbone of our model. As such, we name the resulting model EM-GCN*. The sample space Λ spans the range $[1, 10]$Å and $\widehat{\Lambda}$ is the discretized space with 0.05 increments.

B.2. Experimental Results

In Table 6, the average test root mean square error (rmse) over ten runs with standard deviation is reported for the graph regression task, where the molecular properties of molecular graphs are to be predicted. The result shown is for the case of $q(\cdot) = p_t(\cdot)$. We observe that EM-GCN* frequently performed better than the baselines. This may be due to the training process of EM-GCN*, which exposes it to diverse graph realizations, allowing it to capture non-covalent interactions that are critical for characterizing the physical properties of molecules. In contrast, the baselines employ the conventional molecular graph representation. We note that our framework does not explicitly incorporate bond angles but it does expose the model to graphs with a broad range of connectivities. This exposure indirectly integrates geometric information, as the latent variable constructs graphs with bond lengths falling within specific ranges. This provides our model with additional 2D information regarding interatomic distances, which may offer insights into the underlying 3D structure.

C. Datasets and Implementation Details

All datasets used in this paper are publicly available and open-source.

C.1. Heterogeneous Datasets

The characteristics of the three heterogeneous benchmark datasets are as summarized in Table 7. In the DBLP `https://dblp.uni-trier.de/` dataset, the research areas of the authors are to be predicted. Meanwhile, in the ACM `http://dl.acm.org/` and IMDB `https://www.imdb.com/interfaces/` datasets, the categories of papers and genres of movies are to be determined, respectively. The datasets are publicly available at `https://github.com/seongjunyun/Graph_Transformer_Networks`.

In the DBLP dataset, there are three distinct node types (Paper (P), Author (A), Conference (C)) and two edge types (PA, PC). The ACM dataset also comprises three node types (Paper (P), Author (A), and Subject (S)) and two edge types (PA, PS).

[2]https://moleculenet.org/datasets-1

[3]A database of purchasable drug-like compounds; https://zinc.docking.org/tranches/home/

Table 7. Heterogeneous datasets. The number of A-B edges is equal to the number of B-A edges thus omitted.

	Edge types (A-B)	# of A-B	# Edges	# Edge types	# Features
DBLP	Paper - Author Paper - Conference	19645 14328	67946	4	334
ACM	Paper - Author Paper - Subject	9936 3025	25922	4	1902
IMDB	Movie - Director Movie - Actor	4661 13983	37288	4	1256

C.2. Homogenous Datasets

The utilized homogeneous graph datasets include Cora, Citeseer, Pubmed, Texas, Cornell and Wisconsin. The first three mentioned datasets citation networks where the nodes represent documents and the edges denote citation linkages (Kipf & Welling, 2017). On the other hand, the latter three datasets are heterophilic, featuring connections between dissimilar nodes. We consider the edges as undirected. The statistics and parameters of the datasets are summarized in Table 8.

Table 8. Statistics and parameters of homogeneous graph datasets.

	# Nodes	# Edges	# Features	# Classes	Threshold τ
Cora	2708	5429	1433	7	0.5
Citeseer	3327	4732	3703	6	0.6
Pubmed	19717	44338	500	3	0.9
Texas	183	295	1703	5	0.6
Cornell	183	280	1703	5	0.6
Wisconsin	251	466	1703	5	0.6

C.3. Chemical Datasets

The selected datasets, FreeSolv, ESOL, and Lipophilicity, are intended for graph regression tasks. In FreeSolv, the task involves estimating solvation energy. In ESOL, the objective is to estimate solubility. Lastly, Lipophilicity is for the estimation of lipophilicity. These are important molecular properties in the realm of physical chemistry and provide insights into how molecules interact with solvents.

- FreeSolv. The dataset contains experimental and calculated hydration-free energy of 642 small neural molecules in water.
- ESOL (for *E*stimated *SOL*ubility). ESOL consists of water solubility data for 1128 compounds.
- Lipophilicity. This dataset contains experimental results of the octanol/water distribution coefficient of 4200 compounds, namely logP at pH 7.4, where P refers to the partition coefficient. Lipophilicity refers to the ability of a compound to dissolve in fats, oils and lipids.

C.4. Hyperparameters, Code and Implementation Tips

The models were trained on a server equipped with four NVIDIA RTX A5000 GPUs for hardware acceleration. As the parameter spaces are small for the datasets used in the paper, we apply the standard Metropolis-Hastings algorithm for MCMC. The code is available at `https://github.com/amblee0306/EMGNN.git`.

Heterogeneous node classification. The Simple-HGN model was obtained from the CogDL library (Cen et al., 2023), SeHGNN from their code repository (`https://github.com/ICT-GIMLab/SeHGNN`). Meanwhile, GAT was sourced from the DGL library (Wang et al., 2019) and GCN is from `https://github.com/tkipf/pygcn`.

The hidden units are set to 64 for all models. The hyperparameters of SeHGNN and Simple-HGN are as in the respective repository. For fair comparison, GAT and GCN are evaluated on the entire graph ignoring the types. Specifically for our model, the number of MCMC iterations N_{mc} is set to be 15000, the T denoting the number of EM iterations and T' is tuned by searching on the following search spaces: $[10, 15, 20, 25, 30]$.

Homogeneous node classification. The BGCN model was obtained from its code repository at `https://github.com/huawei-noah/BGCN/tree/master`. Likewise, for RSGNN, the model was sourced from its code repository located at `https://github.com/EnyanDai/RSGNN`. As for the ACM-GCN+ model, it was acquired from its code repository (`https://github.com/SitaoLuan/ACM-GNN`). The hyperparameters used for evaluating the models align with those specified in their respective repositories if provided.

Unlike heterogeneous graphs, during implementation, once $\boldsymbol{\lambda} = (\lambda_1, \lambda_2)$ is determined in MCMC, we still need to draw a graph $G_{\boldsymbol{\lambda}}$ according to $\boldsymbol{\lambda}$. The graph $G_{\boldsymbol{\lambda}}$ is stored for later use.

In the context of our model, N_{mc} is set to be 15000, the T and T' is tuned by searching on the following search spaces: $[10, 15, 20, 25, 30]$. The choice of the threshold τ (see Section 5.2) is tabulated in Table 8. Notice that the threshold τ is chosen such that E' for each dataset is neither too small or too big.

Graph regression on molecular datasets. The experiments on this task were facilitated using the DGL-LifeSci library (Li et al., 2021b) which provided the code and hyperparameters for the baseline models. For fair comparisons, all the models were evaluated using the canonical features as documented at `https://lifesci.dgl.ai/generated/dgllife.utils.CanonicalAtomFeaturizer.html`. Specifically for our model, N_{mc} is set to 18000, T and T' is tuned by searching on the following search spaces: $[10, 15, 20, 25, 30]$.

D. Complexity

We discuss the complexity of EM-GCN, focusing on the steps involving MCMC. The complexity of the remaining steps depends mainly on the chosen base model, e.g., GCN. We mainly compare with BGCN and RSGNN, which share certain common features with our model.

In the E-step, where MCMC is executed, the model parameters remain fixed. Identical inputs to the "frozen" model consistently produce the same $L_{\mathbf{X}}(\boldsymbol{\lambda}, \boldsymbol{\theta})$. Hence, to expedite MCMC iterations for a specific $\boldsymbol{\theta}$, we precompute, store and reuse $L_{\mathbf{X}}(\boldsymbol{\lambda}, \boldsymbol{\theta}), G_{\boldsymbol{\lambda}}$ for all $\boldsymbol{\lambda} \in \Lambda$, using a space complexity of $O(|\Lambda|(|E| + |V|))$, compared to BGCN's $O(|E| + |V|)$ and RSGNN's $O(|V|^2)$. Therefore, the space complexity of EM-GCN is higher than that of BGCN in general. It is lower than RSGNN for sparse graphs if $|V| \gg |\Lambda|$, i.e., the size of the (discretized) sample space is much smaller than the number of nodes.

As such, our model takes a comparatively lower training time than RSGNN and BGCN. The theoretical time complexity of our model is $O(K|\Lambda|\cdot|E|+N_{mc})$ for N_{mc} MCMC iterations and K GCN layers (we consider only the message passing operation in GCN layers). On the other hand, the time complexity of RSGNN with κ MLP layers is $O(|V|d'd + (\kappa - 1)Nd^2 + K|E|)$ and BGCN is $O(K|E| + N_b|V|^2 N_c)$ for N_b MMSBM iterations, N_c classes, d' feature size and d hidden dimension. Time complexity does not include extra time taken for backpropagation in models with more parameters like BGCN and RSGNN.

We show the explicit run-times (in seconds) for the models compared in Table 9, with the same software and hardware configurations. We see that EM-GCN is much faster than both BGCN and RSGNN.

Table 9. Run-time (in sec.) comparison among different models

	Cora	Citeseer	Pubmed
BGCN	328.14 ± 2.18	323.55 ± 4.86	1446.91 ± 31.31
RSGNN	147.09 ± 4.68	164.83 ± 1.71	341.52 ± 3.06
EM-GCN	86.06 ± 1.39	65.19 ± 14.27	102.18 ± 2.63

E. Node-level Out-of-distribution Detection

To demonstrate the versatility of our model, we apply it to OOD detection tasks. We specifically target two types of OOD nodes: nodes with random feature interpolation and nodes from Left-Out classes. Our experimental configurations remain consistent with those outlined in (Wu et al., 2023). Although our model is not explicitly designed for node-level OOD detection (as there are no specific elements in our approach targeting features or labels), we observe that it has competitive performance compared to most benchmarks, as evidenced by its AUROC, AUPR, and FPR95 scores. Moreover, our model achieves results comparable to the top-performing baseline, GNNSAFE (Wu et al., 2023). Our model also demonstrates strong performance on in-distribution (ID) data, which aligns with the objective of the OOD task: to identify OOD test samples without sacrificing accuracy within the ID data.

We also apply a simple modification of EM-GCN to integrate it with the Energy-Based Model (EBM). Specifically, we utilize the same output logits from EM-GCN and incorporate them into equation (5) of (Wu et al., 2023) to compute the energy score for OOD detection during inference. This modification results in improvements on the Leave-out classes task, surpassing the performance of GNNSAFE. However, similar enhancements are not observed in the case of random features, which could be an area of investigation in the future. Experimental results can be seen in Table 10. For EMGNN, the GCN backbone is employed, thus named EM-GCN.

Table 10. OOD detection task on Cora dataset. Performance measured by AUROC/AUPR/FPR95. For AUROC and AUPR, higher values indicate better performance, while for FPR95, lower values are better. In-distribution testing accuracy (ID-ACC) is reported for calibration.

	Cora (Random Feature)				Cora (Left-Out Classes)			
Model	**AUROC(↑)**	**AUPR(↑)**	**FPR95(↓)**	**ID-ACC**	**AUROC(↑)**	**AUPR(↑)**	**FPR95(↓)**	**ID-ACC**
MSP	85.39	73.70	64.88	75.30	91.36	78.03	34.99	88.92
ODIN	49.88	26.96	100.00	75.00	49.80	24.27	100.00	88.92
Mahalonabis	49.93	31.95	99.93	74.90	67.62	42.31	90.77	88.92
Energy	86.15	74.42	65.81	76.10	91.40	78.14	41.08	88.92
GKDE	82.79	66.52	68.24	74.80	57.23	27.50	88.95	89.87
GPN	85.88	73.79	56.17	<u>77.00</u>	90.34	77.30	**27.42**	**91.46**
GNNSAFE	**93.18**	**87.83**	<u>44.21</u>	75.76	<u>92.92</u>	<u>82.46</u>	31.49	89.15
EM-GCN	<u>92.09</u>	<u>83.15</u>	**37.87**	**82.33**	90.24	81.44	51.89	<u>91.42</u>
EM-GCN (EBM)	-	-	-	-	**93.43**	**83.56**	<u>29.24</u>	90.91

F. Related Work

This section provides an overview of related work, specifically on the selected baseline models and their relevance to our model. On the aspect of heterogeneous graphs, GTN (Yun et al., 2019) emphasized that disregarding the type information of such graphs and treating them as homogeneous is suboptimal. To address this, they devised a method to learn new graph structures using multiple candidate (edge type-specific) adjacency matrices and attention mechanisms. Their work is relevant to ours because, akin to GTN, we consider the given graph suboptimal. However, instead of learning new meta-path-related graph structures that introduce connections between unconnected nodes in the original graph, our setup and approach focus on the information transmission rates in such graphs.

On the other hand, concerning homogeneous graphs, given that our model focuses on learning from a distribution of graphs, it makes sense to compare against methods that alter the observed one, albeit in different ways. As such, DropEdge (Rong et al., 2020) and RSGNN (Dai et al., 2022) were considered. In the case of DropEdge, edges in the given graph were randomly dropped at a manually selected rate in each epoch. This process generated different copies of the original graph, diversifying the input data used to train the model. Consequently, DropEdge was deemed capable of overcoming overfitting and oversmoothing.

Meanwhile, for RSGNN, the model trained a link predictor to rewire the observed graph, eliminating or down-weighting noisy edges (connecting nodes with dissimilar features) and adding edges to densify the graphs. This was done to ensure the model is robust to noise and to address the issue of label sparsity by connecting more unlabeled nodes to labeled ones. In (Ji et al., 2023c), a two-stage scheme is proposed to enhance the performance of existing GNN models, where the second stage involves edge drop along estimated class boundaries.

Our work also shares a close connection with BGCN (Zhang et al., 2019), a work employing a Bayesian formulation tailored to address uncertainty in graphs. In their framework, the model parameters were treated as random variables, incorporating a prior distribution. Additionally, BGCN interpreted the observed graph as an instantiation from a parametric family of random graphs. Specifically, it adopted the assortative mixed membership stochastic block model (aMMSBM) as the underlying graph model.

G. Theoretical Discussions

G.1. Discussions and the Proof of Theorem 1

Assume the parameter sample space Λ is a metric space, whose metric is $d(\cdot,\cdot)$. Define the *Wasserstein space* $\mathcal{P}(\Lambda)$ to be the space of (Borel) probability distributions on X with finite mean and variance.

Given μ_1, μ_2 in $\mathcal{P}(\Lambda)$, the *Wasserstein metric* $W(\mu_1, \mu_2)$ is defined by

$$W(\mu_1,\mu_2)^2 = \inf_{\gamma\in\Gamma(\mu_1,\mu_2)} \int d(\lambda,\lambda')^2 \,\mathrm{d}\gamma(\lambda,\lambda'),$$

where $\Gamma(\mu_1,\mu_2)$ is the set of *couplings* of μ_1, μ_2, i.e., the collection of probability measures on $\Lambda\times\Lambda$ whose marginals are μ_1 and μ_2, respectively.

Intuitively, the Wasserstein metric is the minimum amount of "work" required to transform one probability distribution into the other, where the "work" is the sum of the product of the amount of probability mass to be moved and the distance that it must be moved. It is well-known that $W(\cdot,\cdot)$ makes $\mathcal{P}(\Lambda)$ a metric space (Villani, 2009; Ji et al., 2023a;b).

Recall that for each $\lambda\in\Lambda$ there is an associated graph shift operator $S_\lambda \in M_n(\mathbb{R})$, where n is the size of the node set V. This means that we have a *parameterization map*

$$\mathfrak{p}: \Lambda \to M_n(\mathbb{R}), \lambda \mapsto S_\lambda,$$

and $M_n(\mathbb{R})$ is endowed with the operator norm denoted by $\|\cdot\|$.

We analyze the following fundamental GNN model. For each graph shift operator S_λ, it defines a GNN layer according to (1), where the aggregation "AGGR" is achieved by multiplying the input features by S_λ. We further assume that the matrix W^k in (1) depends only on $\boldsymbol{\theta}$ and the "ReLU" function is used for the activation σ. For a K, let $\Psi(\lambda,\cdot;\boldsymbol{\theta})$ be the resulting K-layer GNN, where $\boldsymbol{\theta}$ is the estimated parameters of the model. For any input features $\mathbf{x}$, we have the expected output $\mathbf{z}_\mu = \mathbb{E}_{\lambda\in\mu}[\Psi(\lambda,\mathbf{x};\boldsymbol{\theta})]$. Therefore, assuming $\mathbf{x},\boldsymbol{\theta}$ are fixed, we have the following *feature map*

$$\mathfrak{f}: \mathcal{P}(\Lambda) \to \mathbb{R}^n, \mu \mapsto \mathbf{z}_\mu = \mathbb{E}_{\lambda\in\mu}[\Psi(\lambda,\mathbf{x};\boldsymbol{\theta})].$$

In general, if $\phi: X_1 \to X_2$ is a map between metric spaces with respective metrics $d_1(\cdot,\cdot)$ and $d_2(\cdot,\cdot)$, then it is called *Hölder continuous* if for some constants $C,\alpha>0$, we have $d_2\big(\phi(x_1),\phi(x_2)\big) \le C d_1(x_1,x_2)^\alpha, x,y\in X_1$. Ignoring the scalar C, we may also say that ϕ is *α-Hölder continuous* to emphasize the exponent.

We can now state the following precise version of Theorem 1.

Theorem 3. *Let K be the number of GNN layers. If the parameterization $\mathfrak{p}:\Lambda\to M_n(\mathbb{R})$ is α-Hölder continuous and bounded, then the feature map $\mathfrak{f}:\mathcal{P}(\Lambda)\to\mathbb{R}^n$ is β-Hölder continuous for $\beta = 2K\alpha/(K\alpha+2)$. Moreover, if the activation σ is bounded, e.g.,* $\tanh$, *sigmoid, then we can remove the condition that $\mathfrak{p}$ is bounded.*

Proof. As $\mathfrak{p}$ is Hölder continuous, there is $C,\alpha>0$ such that $\|S_\lambda - S_{\lambda'}\| \le C d(\lambda,\lambda')^\alpha$. Moreover, by the assumption, there is an upper bound B on the norms of the image of $\mathfrak{p}$.

For each $\lambda\in\Lambda$, the k-th GNN layer for $k\le K$ takes the form $\mathbf{y}\mapsto\sigma(S_\lambda\mathbf{y}W^k)$, where W^k is determined by the fixed model parameters $\boldsymbol{\theta}$. As the activation function σ is 1-Lipshitz, we have

$$\|\sigma(S_\lambda\mathbf{y}W^k) - \sigma(S_{\lambda'}\mathbf{y}W^k)\| \le \|W^k\|\|\mathbf{y}\|\|S_\lambda - S_{\lambda'}\| \le C_1\|\mathbf{y}\|d(\lambda,\lambda')^\alpha, \tag{12}$$

where C_1 is the constant $C\|W^k\|$. Taking $\mathbf{y}$ as the output of the previous layer, we may repeat (12) and obtain

$$\|\Psi(\lambda,\mathbf{x};\boldsymbol{\theta}) - \Psi(\lambda',\mathbf{x};\boldsymbol{\theta})\| \le C_2 d(\lambda,\lambda')^{K\alpha}, \tag{13}$$

where C_2 is a constant independent of λ, λ'.

Moreover, due to the norm upper bound B of S_λ for any $\lambda \in \Lambda$, there is an upper bound B_1 of $\Psi(\lambda, \mathbf{x}; \boldsymbol{\theta})$ independent of λ. If the activation σ is bounded, then the same boundedness conclusion holds without assuming $\mathfrak{p}$ is bounded as the last layer ends with the bounded function σ.

To prove the theorem, consider $\mu_1, \mu_2 \in \mathcal{P}(\Lambda)$. Let $\gamma \in \Gamma(\mu_1, \mu_2)$ be a distribution on $\Lambda \times \Lambda$ such that

$$W(\mu_1, \mu_2)^2 = \int d(\lambda, \lambda')^2 \,\mathrm{d}\gamma(\lambda, \lambda'). \tag{14}$$

As the marginals of γ are μ_1 and μ_2 respectively, we may express $\|\mathfrak{f}(\mu_1) - \mathfrak{f}(\mu_2)\|$ as

$$\begin{aligned}
&\|\mathfrak{f}(\mu_1) - \mathfrak{f}(\mu_2)\| = \|\mathbb{E}_{\lambda \in \mu_1}[\Psi(\lambda, \mathbf{x}; \boldsymbol{\theta})] - \mathbb{E}_{\lambda \in \mu_2}[\Psi(\lambda, \mathbf{x}; \boldsymbol{\theta})]\| \\
=&\|\int \Psi(\lambda, \mathbf{x}; \boldsymbol{\theta}) \,\mathrm{d}\mu_1(\lambda) - \int \Psi(\lambda', \mathbf{x}; \boldsymbol{\theta}) \,\mathrm{d}\mu_2(\lambda')\| \\
=&\|\int \Psi(\lambda, \mathbf{x}; \boldsymbol{\theta}) - \Psi(\lambda', \mathbf{x}; \boldsymbol{\theta}) \,\mathrm{d}\gamma(\lambda, \lambda')\|.
\end{aligned}$$

Let $a > 0$ be a number to be determined later. By (14), the Chebyshev inequality implies that the γ-measure of the set $\Sigma_a = \{(\lambda, \lambda') \mid d(\lambda, \lambda') \geq a\}$ is bounded by $W(\mu_1, \mu_2)/a^2$. With this, we estimate $\|\mathfrak{f}(\mu_1) - \mathfrak{f}(\mu_2)\|$ as follows

$$\begin{aligned}
&\|\mathfrak{f}(\mu_1) - \mathfrak{f}(\mu_2)\| \leq \int \|\Psi(\lambda, \mathbf{x}; \boldsymbol{\theta}) - \Psi(\lambda', \mathbf{x}; \boldsymbol{\theta})\| \,\mathrm{d}\gamma(\lambda, \lambda') \\
=& \int_{\Sigma_a} \|\Psi(\lambda, \mathbf{x}; \boldsymbol{\theta}) - \Psi(\lambda', \mathbf{x}; \boldsymbol{\theta})\| \,\mathrm{d}\gamma(\lambda, \lambda') + \int_{\Lambda \times \Lambda \backslash \Sigma_a} \|\Psi(\lambda, \mathbf{x}; \boldsymbol{\theta}) - \Psi(\lambda', \mathbf{x}; \boldsymbol{\theta})\| \,\mathrm{d}\gamma(\lambda, \lambda') \\
\leq& \int_{\Sigma_a} 2B_1 \,\mathrm{d}\gamma(\lambda, \lambda') + \int_{\Lambda \times \Lambda \backslash \Sigma_a} C_2 d(\lambda, \lambda')^{K\alpha} \,\mathrm{d}\gamma(\lambda, \lambda') \\
\leq& \frac{2B_1 W(\mu_1, \mu_2)^2}{a^2} + C_2 a^{K\alpha}.
\end{aligned}$$

Minimizing the right-hand-side (by taking its derivative w.r.t. a), we have $a = C_3 W(\mu_1, \mu_2)^{2/(K\alpha+2)}$, where $C_3 = [4B_1/(C_2 K\alpha)]^{1/(K\alpha+2)}$. Plugging in the expression of a into the estimation, we have

$$\|\mathfrak{f}(\mu_1) - \mathfrak{f}(\mu_2)\| \leq \frac{2B_1}{C_3^2} W(\mu_1, \mu_2)^{2-4/(K\alpha+2)} + C_2 C_3^{K\alpha} W(\mu_1, \mu_2)^{2K\alpha/(K\alpha+2)}.$$

Therefore, $\mathfrak{f}$ is β-Hölder continuous for $\beta = 2K\alpha/(K\alpha + 2)$. □

G.2. The Proof of Theorem 2

The proof is a refinement of the proof of (Hardt et al., 2016, Theorem 3.12). As $L_{\mathbf{X}}$ is assumed to be convex, we observe that in the expression for $J_{\Lambda_{T'}}(\widehat{\boldsymbol{\theta}})$, a term is convex if $p(\lambda_i) \geq p_0(\lambda_i)$ and concave otherwise. Therefore, we need to separate these two cases when performing gradient descent.

We follow the proof of (Hardt et al., 2016, Theorem 3.12), and highlight necessary changes. By (Hardt et al., 2016, Theorem 2.2), it suffices to show that the algorithm $\mathcal{A}$ is ϵ-uniformly stable (Hardt et al., 2016, Definition 2.1). Let $\Lambda_1 = \{\lambda_{1,1}, \ldots, \lambda_{1,T'}\}$ and $\Lambda_2 = \{\lambda_{2,1}, \ldots, \lambda_{2,T'}\}$ be two sample sequences of λ that differ in only a single sample. Consider the gradient updates $\Gamma_{1,1}, \ldots, \Gamma_{1,T'}$ and $\Gamma_{2,1}, \ldots, \Gamma_{2,T'}$. Let $\widehat{\boldsymbol{\theta}}_{1,t'}$ and $\widehat{\boldsymbol{\theta}}_{2,t'}, t' \leq T'$ be the corresponding outputs of the algorithm $\mathcal{A}$.

Introduce $f(\lambda, \boldsymbol{\theta}) = \frac{p(\lambda) - p_0(\lambda)}{q(\lambda)} L_{\mathbf{X}}(\lambda, \boldsymbol{\theta})$. As $L_{\mathbf{X}}(\lambda, \cdot)$ is convex, α-Lipschitz and β-smooth, $f(\lambda, \boldsymbol{\theta})$ is $\alpha\gamma$-Lipschitz, $\beta\gamma$-smooth. Moreover, it is convex if $p(\lambda) \geq p_0(\lambda)$.

Write $\delta_{t'}$ for $\|\widehat{\boldsymbol{\theta}}_{1,t'} - \widehat{\boldsymbol{\theta}}_{2,t'}\|$. Using the fact that $f(\lambda, \cdot)$ is $(\alpha\gamma)$-Lipschitz, by (Hardt et al., 2016, Lemma 3.11), we have for any $t_0 \leq T'$

$$\begin{aligned}
&\mathbb{E}\big(|f(\lambda, \widehat{\boldsymbol{\theta}}_{1,T'}) - f(\lambda, \widehat{\boldsymbol{\theta}}_{2,T'})|\big) \\
\leq& \frac{t_0}{T'} + \alpha\gamma \mathbb{E}(\delta_{T'} \mid \delta_{t_0} = 0).
\end{aligned}$$

We need an estimation of $\mathbb{E}(\delta_{T'} \mid \delta_{t_0} = 0)$. For convenience, for any $t' \geq t_0$, let $\Delta_{t'} = \mathbb{E}(\delta_{t'} \mid \delta_{t_0} = 0)$.

Observe that at step t', with probability $1 - 1/n$, the samples selected are the same in both Λ_1 and Λ_2. Moreover, with probability $(1 - 1/n)b_1$, the common sample is convex, so we can apply (Hardt et al., 2016, Lemma 3.7.2). With probability $(1 - 1/n)(1 - b_1)$, the common sample is non-convex, and (Hardt et al., 2016, Lemma 3.7.1) is applicable. With probability $1/n$, the selected samples are different and $\Gamma_{1,t'}$ and $\Gamma_{2,t'}$ are respectively $\frac{|p(\lambda)-p_0(\lambda)|}{q(\lambda)}\alpha a_{t'}$-bounded, by (Hardt et al., 2016, Lemma 3.3).

Therefore, by linearity of expectation and (Hardt et al., 2016, Lemma 2.5), we may estimate:

$$\begin{aligned}
\Delta_{t+1} &\leq (1 - \frac{1}{n})\big(b_1 + (1 - b_1)(1 + a_{t'}\beta\gamma)\big)\Delta_{t'} + \frac{1}{n}\Delta_{t'} \\
&+ \frac{\alpha a_{t'}}{n}\mathbb{E}_{\lambda_{1,t'},\lambda_{2,t'}}\big(\frac{|p(\lambda_{1,t'}) - p_0(\lambda_{1,t'})|}{q(\lambda_{1,t'})} \\
&+ \frac{|p(\lambda_{2,t'}) - p_0(\lambda_{2,t'})|}{q(\lambda_{2,t'})}\big) \\
&= (1 - \frac{1}{n})\big(b_1 + (1 - b_1)(1 + a_{t'}\beta\gamma)\big)\Delta_{t'} \\
&+ \frac{1}{n}\Delta_{t'} + \frac{2b_2\alpha a_{t'}}{n} \\
&\leq (1 - \frac{1}{n})\big(b_1 + (1 - b_1)(1 + \frac{c\beta\gamma}{t'})\big)\Delta_{t'} \\
&+ \frac{1}{n}\Delta_{t'} + \frac{2b_2\alpha c}{nt'} \\
&= \big(1 + (1 - \frac{1}{n})(1 - b_1)\frac{c\beta\gamma}{t'}\big)\Delta_{t'} + \frac{2c(b_2\alpha)}{nt'} \\
&\leq \exp\big((1 - \frac{1}{n})(1 - b_1)\frac{c\beta\gamma}{t'}\big)\Delta_{t'} + \frac{2c(b_2\alpha)}{nt'} \\
&= \exp\Big((1 - \frac{1}{n})\frac{c\big((1 - b_1)\beta\gamma\big)}{t'}\Big)\Delta_{t'} + \frac{2c(b_2\alpha)}{nt'}.
\end{aligned}$$

Then, by the same argument as in the proof of (Hardt et al., 2016, Theorem 3.12), we have the algorithm $\mathcal{A}$ is ϵ-uniformly stable for any

$$\epsilon \leq C\big(\frac{b_2^2\alpha^2}{T'}\big)^{\frac{1}{\beta\gamma c(1-b_1)+1}},$$

for some constant C independent of T' and α.

H. Miscellaneous Discussions

H.1. The Intuition of Typicality (cf. (6))

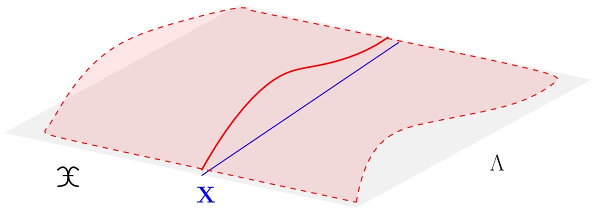


Figure 5. An illustration of the typicality assumption.

As illustrated in Fig. 5, we are interested in taking the average, formally the expectation w.r.t. a distribution π_0, of a function over $\Lambda \times \mathfrak{X}$. The function is illustrated by the red surface in Fig. 5. The "typicality" assumption on $\mathbf{X} \in \mathfrak{X}$ requires that there is a distribution $p_{0,\mathbf{X}}$ such that the above average over $\Lambda \times \mathfrak{X}$ is (approximately) the same as the average, i.e., expectation w.r.t. $p_{0,\mathbf{X}}$, over the "line" $\Lambda \times \{\mathbf{X}\}$. Intuitively, "typicality" says that the cross-section of the function at $\mathbf{X}$ displays the same pattern as the function on the entire domain $\Lambda \times \mathfrak{X}$. Hence, statistics on the entire domain can be estimated from observations on the cross-section $\Lambda \times \{\mathbf{X}\}$.

As we have mentioned in Remark 1, "typicality" allows us to simplify the optimization problem. For future work, we are also interested in solving the optimization with a weaker assumption.

H.2. The Dominant Component of (9)

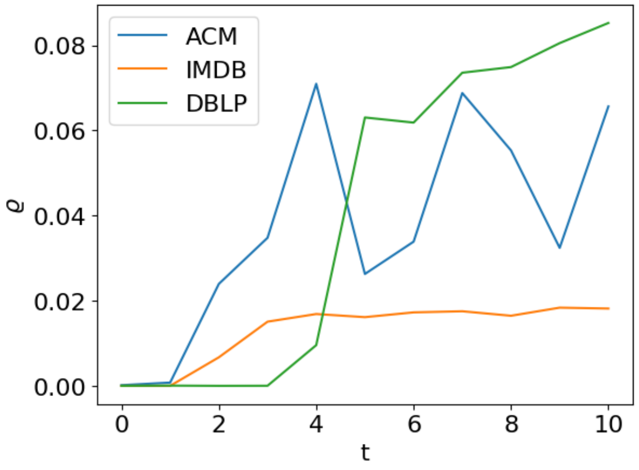


Figure 6. Plot of ϱ across t EM iterations

In Section 4.1, we use $-\eta^{(t)}\mathbb{E}_{\lambda\sim p_0}\big(L_{\mathbf{X}}(\lambda,\boldsymbol{\theta})\big)$ to approximate $\log C(\boldsymbol{\theta})$. To justify this, we provide numerical evidence that the dominant component of (9) is $-\eta^{(t)}\mathbb{E}_{\lambda\sim p_0}\big(L_{\mathbf{X}}(\lambda,\boldsymbol{\theta})\big)$, based on Section 5.1. The above assertion is supported by assessing the ratio

$$\varrho = \frac{\rho}{-\eta^{(t)}\mathbb{E}_{\lambda\sim p_0}\big(L_{\mathbf{X}}(\lambda,\boldsymbol{\theta})\big)},$$

where $\rho = \frac{\mathrm{var}\big(\exp(-\eta^{(t)}L_{\mathbf{X}}(\lambda,\boldsymbol{\theta}))\big)}{2\big(\mathbb{E}_{\lambda\sim p_0}\exp(-\eta^{(t)}L_{\mathbf{X}}(\lambda,\boldsymbol{\theta}))\big)^2}$.

We plot the ϱ values for the heterogeneous graph datasets on EM-GCN[uut] over multiple t iterations in Figure 6. We found that ϱ consistently exhibits small absolute values, supporting the postulation that $-\eta^{(t)}\mathbb{E}_{\lambda\sim p_0}\big(L_{\mathbf{X}}(\lambda,\boldsymbol{\theta})\big)$ is the main component in (9).