Graph Decoupling Attention Markov Networks for Semisupervised Graph Node Classification

Jie Chen, Shouzhen Chen, Mingyuan Bai, Jian Pu, Member, IEEE, Junping Zhang, Senior Member, IEEE, and Junbin Gao

Abstract—Graph neural networks (GNNs) have been ubiquitous in graph node classification tasks. Most GNN methods update the node embedding iteratively by aggregating its neighbors’ information. However, they often suffer from negative disturbances, due to edges connecting nodes with different labels. One approach to alleviate this negative disturbance is to use attention to learn the weights of aggregation, but current attention-based GNNs only consider feature similarity and suffer from the lack of supervision. In this article, we consider label dependency of graph nodes and propose a decoupling attention mechanism to learn both hard and soft attention. The hard attention is learned on labels for a refined graph structure with fewer interclass edges so that the aggregation’s negative disturbance can be reduced. The soft attention aims to learn the aggregation weights based on features over the refined graph structure to enhance information gains during message passing. Particularly, we formulate our model under the expectation–maximization (EM) framework, and the learned attention is used to guide label propagation in the M-step and feature propagation in the E-step, respectively. Extensive experiments are performed on six well-known benchmark graph datasets to verify the effectiveness of the proposed method.

Index Terms—Deep learning, graph convolutional networks, network representation learning.

I. INTRODUCTION

NODE representation learning on graphs aims to extract high-level features from the node and its neighborhood. It has been proven useful for research areas, such as social influence [1], [2], knowledge graphs [3], [4], chemistry and biology [5], and recommendation systems [6], [7]. Recently, various graph neural networks (GNNs) [8]–[10] have emerged to solve the node classification problem and achieve state-of-the-art results. Most of them are formulated under the message passing framework [11]. Each node passes and aggregates messages to/from its neighbors according to edges to achieve information gain and update its embedding [12]–[14].

However, information gain of message passing is not always beneficial in the node classification task because edges of real-world graphs often connect nodes with different labels. According to [15], graph convolution networks (GCNs) are regarded as Laplacian smoothers on features. Repeatedly applying the graph convolution operator may smooth the features of the same-label vertices if they are densely connected under the graph structure. Hence, the difficulty of the subsequent classification task is significantly reduced. However, the graph structure can be noisy from many interclass edges (which connect two nodes with different labels) [14], [16]. In this case, the graph convolution operator may introduce negative disturbances that arise from neighbors with different labels [14], [16], [17]. The reason is that it smooths the features with different labels and blurs the classification boundary. Thus, for the node classification task, simply aggregating features under the original graph structure often cannot achieve the optimal performance [14]. Taking account of different contributions from the nodes in a graph is important, as not all the edges have equal impacts. A wiser solution is to learn the edge weights for message passing in the training stage. Furthermore, labels’ information that influences the aggregation quality should also be considered in the learning process.

One approach to alleviate negative disturbance is to leverage the attention mechanism [14], [18]. It learns the aggregation weights of edges based on features to choose the critical message and guide feature propagation [9]. However, current attention mechanisms on graphs suffer from the lack of supervision [19]. For example, for citation networks, the weights learned by attention are highly dataset-dependent and degenerate to near-uniform [20]. In addition, the attention learned using those methods is usually proportional to feature similarity and omits label dependency. To solve these issues, in this article, we argue that attention learning on graphs should be decoupled into structure learning on labels and edge weight learning on features, where more constraints should be applied to learn meaningful attention patterns.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Based on previous studies, by conducting validation experiments in Fig. 1, we demonstrate how the graph structure and edge weights affect message passing. We first investigate how the prediction performance varies with the ratio of interclass edges in the graph structure. In Fig. 1(a), we observe that with the increment of the ratio of interclass edges, the test accuracy decreases almost monotonically. It confirms that nodes will receive negative information from their interclass neighbors (whose labels are different from these nodes) for classification tasks. Hence, message passing between interclass nodes should be avoided. Next, we examine how performance varies with different aggregation weights. Using vanilla GCN as a baseline, we compare two opposite attention mechanisms, i.e., positive relativity (PR) attention and negative relativity (NR) attention. The PR is proportional to \( \cos(x_i, x_j) \) and the NR is proportional to \( -\cos(x_i, x_j) \), where \( x_i \) and \( x_j \) are the features of nodes \( i \) and \( j \), respectively. As shown in Fig. 1(b), for the results using the oracle adjacency matrix (all the interclass edges are manually removed), NR is evidently better than PR. While using the original adjacency matrix, the trend is reversed, and PR is obviously better than NR. It indicates that NR is more helpful for message passing under the oracle graph because it pays more attention to divergent intraclass neighbors to obtain more information gain. However, when the interclass connection exists in the original graph, negative disturbance is more involved for NR and surpasses its benefits due to the natural difference between different classes.

Thereby, a more flexible attention model for graph node classification consists of two parts, i.e., a refined graph structure reflecting label similarity (which has fewer interclass edges) and reasonable edge weights (which indicates the aggregation weights for message passing). Furthermore, the learning mechanisms of structures and weights are different. To obtain a refined graph structure, we need to use node label information to reduce the interclass edge ratio of the graph. With a refined graph at hand, reasonable weights can be effectively learned by features to avoid negative disturbance and enhance information gain of aggregation.

In this work, we consider label dependency and propose the graph decoupling attention Markov networks (GDAMNs) to decouple the attention learning on the graph for structure learning and weight learning. To learn a better graph structure, we model the uncertainty of edges with variational inference and use the hard attention mechanism on node labels to encode the graph structure as a latent variable. As in the aforementioned validation experiment, we observe that a better graph structure has fewer interclass edges. Instead of directly using the pseudo-labels to identify interclass edges, we treat the distribution of node labels as latent variables and evaluate the pseudo-labels iteratively under the expectation–maximization (EM) framework. Furthermore, we impose a novel graph structure prior based on labels and update it in each EM iteration to assist hard attention learning. To learn better weights, we apply the soft attention mechanism on node features over the refined graph structure. We also propose a mutual information constraint as extra supervision in soft attention learning to learn useful weights and enhance information gain. Moreover, the learned attention guides label propagation in the M-step and feature propagation in the E-step.

The contributions of this work are summarized as follows.

1) We decouple the attention learning procedure on the graph to a hard attention graph structure learning on labels and a soft attention graph weight learning on features. We propose to learn a better attention pattern with the constraints that assist in maximizing information gain.

2) We define a novel latent graph structure prior over label distribution and use variational inference to obtain a better graph structure with fewer interclass connections.

3) We formulate the proposed method under the EM framework. The better graph structure and aggregation weights learned in M-step label propagation guide feature propagation of the E-step to learn node representation and infer pseudo-labels’ distribution.

4) Extensive experiments are designed and performed to verify the efficiency and superiority of our model and the learned attention.

The remaining part of the article is organized as follows. Section II briefly reviews the related methods. Section III introduces the notations and related graph networks. In Section IV, we present the proposed GDAMN method in detail. Evaluation results on six benchmark datasets and ablation studies are presented in Section V to verify the effectiveness and robustness of the proposed method. The final Section VI concludes this article.

II. RELATED WORK

There is past literature recognizing negative disturbance in message passing. Li et al. [15] proved that the graph convolution operator is actually a special form of Laplacian smoothing. To overcome negative disturbance, they proposed to combine co-training and self-learning methods to train the GCN. Moreover, Hou et al. [14] discovered that the neighborhood will provide both positive information and negative disturbance for a given task during propagation. They proposed smoothness metrics to selectively aggregate...
neighborhood information to amplify useful information and reduce negative disturbance over the original graph. In addition, Xie et al. [17] found that neighborhood aggregation may be harmful or unnecessary in some cases. If a node’s neighbors have high entropy, further aggregation will jeopardize model performance, and any more aggregation is unnecessary if a node is nearly identical to its neighbors. However, these methods do not show that the graph structure and weights may have different influences on message passing. Unlike them, we decouple the graph structure and influence of weights on message passing and use different learning mechanisms to overcome negative disturbance and enhance information gain.

To improve the effectiveness of message passing, there are attempts combining learnable attention functions to assign an importance weight to every neighboring node according to node feature similarity. Graph attention networks (GATs) [9] apply multthead self-attention mechanisms to learn aggregation weight distribution on edges among nodes and update node features and achieve remarkable performance. Because of such a successful application of attention mechanisms on GNNs, an in-depth critical study of it becomes a new trend. Li et al. [20] showed that attention learned by GATs is highly dataset-dependent, and distributions across heads and layers are nearly uniform for all citation networks. It is difficult to truly capture effective weights that are helpful to the tasks. Wang et al. [19] indicated that GATs suffer from over-fitting due to the increasing number of parameters and the lack of direct supervision on attention weights. They proposed margin-based constraints to reduce information propagation between nodes belonging to different classes. However, in most current attention-based approaches, the learned aggregation weights are based on node features, and they fail to capture the useful information provided by label dependency. Hence, we instead take both feature and label information to learn better attention patterns and propose an extra mutual information constraint to enhance supervision.

There are other methods that consider label dependency on graphs. Label propagation algorithm (LPA) assumes that two nodes linked by an edge are more likely to have the same label and encourages the prediction distribution to be equal to a weighted sum of its neighbor distributions [21], [22]. Besides, Wang and Leskovec [16] analyzed the theoretical relationship between LPA and GCN in terms of feature/label smoothness and influence. They proposed a GCN-LPA model by applying LPA as regularization on training set labels to learn aggregation weights. Huang et al. [23] used a postprocess mechanism called C&S to learn the residual between the ground-truth labels and the basic model’s prediction. Specifically, they proposed the Autoscale and FDiff-scale strategies to enhance label propagation to correct and smooth the prediction of basic models. However, since these methods only consider label dependency under the original graph structure, they are limited by the number of few known labels under the setting of semisupervised learning. There are other works explicitly using GNNs to propagate labels. Rossi et al. [24] presented an inductive–transductive learning scheme based on GNNs and enriched the node features with the target label in the diffusion process. Graph Markov neural networks (GMNNs) [25] used another GCN to update each node label with labels of its neighbors nonlinearly. The key idea of GMNN is to regard the unlabeled nodes as latent variables and use the EM framework to iteratively infer better pseudo-labels. However, they neglect the fact that a better graph structure is helpful for the propagation process. To the best of our knowledge, the proposed GDAMNs are the first work to simultaneously consider label influence for structures and the feature influence for weights.

III. BACKGROUND

A. Notations and Problem Setting

Let \( G = (\mathcal{V}, \mathcal{E}, \mathbf{X}) \) be a graph with the vertex set \( \mathcal{V} = \{v_1, \ldots, v_N\} \) and \( \mathcal{V} = |\mathcal{V}| \). Let \( v_i, v_j \in \mathcal{V} \) and the edge set is denoted as \( \mathcal{E} \) where \( e_{ij} = (v_i, v_j) \in \mathcal{E} \). Each vertex \( v_i \) corresponds to a \( d \)-dimensional feature representation \( \mathbf{x}_i \in \mathbb{R}^d \) \((i = 1, \ldots, N)\), and we denote \( \mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N] \). Each \( y_i \) corresponds to a \( c \)-dimensional one-hot label representation \( y_i \in \mathbb{R}^c \) for vertex \( v_i \), and we denote \( \mathbf{Y} = [y_1, \ldots, y_N] \).

Given the labels \( \mathbf{Y}_C \) of the nodes \( \mathcal{C} \subset \mathcal{V} \), our goal is to predict the labels \( \mathbf{Y}_U \) of the unlabeled nodes \( \mathcal{U} = \mathcal{V} \setminus \mathcal{C} \) by exploiting the graph structure \( \mathcal{E} \) and the features \( \mathbf{X} \) corresponding to all the nodes.

B. Message Passing in Graph Convolutional Networks

For each node \( v_i \in \mathcal{V} \), we denote \( \mathcal{N}(i) = \{j : A_{i,j} \neq 0\} \) as its neighbor set according to the edge set \( \mathcal{E} \) and the adjacency matrix \( \mathbf{A} \). For the \( h \)-th layer of a GCN, we use \( \mathbf{h}^i_{\ell} \) to represent the embedding of node \( i \), \( (\mathbf{W}^\ell, \mathbf{b}^\ell) \) to denote the weights and the bias, and \( \sigma(\cdot) \) to be the nonlinear activation function. The general GCN message passing rule at the \( h \)-th layer for node \( i \) is usually formulated by two steps:

\[
\mathbf{m}^i_{\ell} = \sum_{j \in \mathcal{N}(i) \setminus i} \alpha_{ij} \mathbf{h}^j_{\ell-1} \quad \text{(Neighborhood aggregate)} \quad (1)
\]

\[
\mathbf{h}^i_{\ell} = \sigma(\mathbf{W}^\ell \mathbf{m}^i_{\ell} + \mathbf{b}^\ell) \quad \text{(Feature transform)} \quad (2)
\]

The aggregation weight \( \alpha_{ij} \) in (1) can be computed using the graph Laplacian [8], [26] or feature-based attention learned in the training process [9], [27] as follows:

\[
\alpha_{ij} = \frac{1}{\sqrt{(\mathcal{N}(i)) + 1} \cdot (\mathcal{N}(j)) + 1}} \quad \text{(Laplacian based)}
\]

\[
\alpha_{ij} = \frac{\exp(\theta_{\mathbf{a}}(\mathbf{h}_i, \mathbf{h}_j))}{\sum_{k \in \mathcal{N}(i)} \exp(\theta_{\mathbf{a}}(\mathbf{h}_i, \mathbf{h}_j))} \quad \text{(Attention based)}
\]

where the function \( \theta_{\mathbf{a}} \) in attention-based GNNs can be a nonlinear transformation to compute the similarity of two node features, i.e., \( \phi_{\mathbf{a}} = \text{LeakyReLU}(\mathbf{a}^T \mathbf{[WH]}_i \parallel \mathbf{WH}_j) \). LeakyReLU [28] is an activation function, \( \parallel \) is concatenation, and \( \mathbf{a} \) is a parameterized vector as in GAT [9].

After repeating the message passing procedure for multiple layers, the useful information can be propagated to each node.
of the entire graph. For a GCN with $L$ layers, the prediction of the unlabeled nodes $i \in \mathcal{U}$ is made using a softmax classifier to the last layer $h_i^L$

$$p(y_i | X) = \text{GCN}(A, X) = \text{Cat}(\text{softmax}(h_i^L))$$

where $\text{Cat}(\cdot)$ operator denotes the categorical distribution for labels.

C. Graph Markov Neural Networks

GMNN [25] uses two GCNs and the pseudo-likelihood variational EM framework to learn label dependency for the semi-supervised node classification task. Because the labels of some nodes are unknown, it is difficult to directly maximize the likelihood of all the nodes $\log p_\theta(Y|X)$, i.e., $\log p_\theta(Y_\mathcal{L}, Y_\mathcal{U}|X)$ which is parameterized by GNN. As a remedy, we compute $\log p_\theta(Y_\mathcal{L}|X)$ which is marginal over the unknown, and evidence lower bound (ELBO) of the log-likelihood function is considered

$$\log p_\theta(Y_\mathcal{L}|X) \geq E_{q_\theta(Y_\mathcal{U}|X)}[\log p_\theta(Y_\mathcal{L}, Y_\mathcal{U}|X) - \log q_\theta(Y_\mathcal{U}|X)]$$

where $q_\theta(Y_\mathcal{U}|X)$ is a variational approximation of the posterior $p_\theta(Y_\mathcal{U}|Y_\mathcal{L}, X)$. The lower bound is alternatively optimized between an M-step and an E-step.

In the M-step, with a fixed variational distribution $q_\theta$, the task is to update a GCN parameterized by $\phi$ by maximizing

$$E_{q_\theta(Y_\mathcal{U}|X)}[\log p_\theta(Y_\mathcal{L}, Y_\mathcal{U}|X)]$$

(5)

Due to complex data relationships, it is hard to define a full joint distribution. Instead, the Markov property is introduced according to the graph local structure. Hence, the likelihood function can be approximated by the following pseudo-likelihood:

$$E_{q_\theta(Y_\mathcal{U}|X)} \sum_{i \in \mathcal{V}} \log p_\theta(y_i|Y_{N(i)}, X)$$

(6)

where $N(i)$ denotes the neighbors of node $v_i$, and $Y_{N(i)}$ represents labels of node $v_i$’s neighbors.

To evaluate the above expectation, we consider the pseudo-label $\hat{Y}$ and take samples for $Y_\mathcal{U}$ according to the current $q_\theta$, and other $\hat{Y}_\mathcal{L}$ remain as the ground-truth labels $Y_\mathcal{L}$ in the training data. Thus, the network $p_\theta$ is optimized by maximizing

$$\sum_{i \in \mathcal{V}} \log p_\theta(\hat{y}_i|\hat{Y}_{N(i)}, X)$$

(7)

Hence, in the M-step of GMNN, the GCN $p_\theta$ can also be seen as a label propagator to reconstruct labels.

In the E-step, a GCN is used to parameterize variational $q_\theta$ for feature propagation according to the predefined structure. The goal is to update $q_\theta$ to approximate the posterior distribution $p_\theta(Y_\mathcal{U}|Y_\mathcal{L}, X)$. Due to complicated relational structures between node labels, exact inference is intractable. Therefore, we approximate it with another variational distribution $q_\theta(Y_\mathcal{U}|X)$. Specifically, based on the mean-field formulation [25], [29] in GMNN, the optimal $q_\theta(y_i|X)$ satisfies

$$q_\theta(y_i|X) \approx p_\theta(y_i|Y_{N(i)}, X) \approx p_\theta(y_i|\hat{Y}_{N(i)}, X)$$

(8)

Moreover, the network $q_\theta$ can be regarded as a feature propagator to produce pseudo-labels.

In this article, we use the EM framework proposed by GMNN to solve the problem of insufficient labels. In subsequence, we can leverage labels’ information to learn the structure in the M-step.

D. Gumbel-Softmax Distribution

It is desired to have the sampling operator differentiable and make the training process of stochastic neural networks end to end. To achieve these goals, the reparameterization trick [30] is a ubiquitous technique with continuous variables. However, when the prior distribution is a discrete random variable, there is no well-defined gradient. The Gumbel-Softmax distribution [31], [32] is such a method to approximate the categorical distributions by a continuous distribution. Then gradients can be calculated via the reparameterization trick easily. The Gumbel-Softmax distribution is defined as

$$\text{Gumbel}(\alpha_{1:K}) = \left( \frac{\exp((\log \alpha_k + G_k)/\tau)}{\sum_{k=1}^K \exp((\log \alpha_k + G_k)/\tau)} \right)_{1:K}$$

where $G_k \sim \text{Gumbel}(0, 1)$.
where the subscript $1:K$ is the index of the random variable vector, and $a_k$ is the $k$th element in this vector and is proportional to the categorical distribution. $G_k$ is noise sampled from the Gumbel distribution. $\tau$ is the temperature parameter that controls Gumbel-Softmax’s “sharpness.” A higher $\tau$ will produce a more uniform one.

**IV. PROPOSED METHOD**

In this section, we present the proposed GDAMNs for semi-supervised graph node classification in detail. Our goal is to decouple the attention learning procedure of message passing into two parts: hard attention on labels for structure learning and soft attention on features for edge weight learning. Under the semisupervised learning setting, there are no sufficient labels for structure learning. To make use of the unlabeled nodes, we treat the unseen label as a latent variable and leverage the variational inference and structure prior for structure learning in the M-step in Section IV-A.1. Then we describe our decoupling attention designed for the graph structure and edge weights’ learning and the mutual information constraint that helps attention learning in the M-step in Section IV-A.2. Finally, we show how to reweight the E-step graph by the aggregation weights learned from the M-step in Section IV-B.

**A. M-Step: Graph Inference and Decoupling Architecture**

In the M-step, we use both the labels and features to learn hard and soft attention. Moreover, for hard attention and structure learning, instead of directly learning the deterministic structure from labels, we propose a novel graph prior to model the uncertainty of edges. Combined with this prior, we use variational inference to take hard attention as the structure encoder to infer better graph structures. For soft attention and weight learning, we impose a neighbor mutual information constraint to learn meaningful weights and thus reduce the effects of negative disturbance.

1) Graph Inference and Label Propagation: In the M-step, the objective is to update label propagation parameters $\phi$ by approximating the E-step’s pseudo-labels. On one hand, similar to GMNN [25], we also use the pseudo-likelihood of Markov networks to approximate the likelihood, as in (7). On the other hand, unlike GMNN [25], we have another goal which is to obtain a new graph according to pseudo-labels’ distribution.

As mentioned in Fig. 1(a), the graph structure and similarities of neighbor labels are highly correlated with information gain during message passing. Hence, instead of performing message passing on labels with the original graph structure as GMNN [25] does, we use label information to learn a better structure. However, pseudo-labels may introduce label noise. Directly using the connected nodes’ label information to determine the existence of edges is usually suboptimal. Therefore, instead of learning a deterministic structure from labels, we use the Bayesian approach to model the uncertainty of the structure and introduce a latent variable for a refined graph structure. In [30], the notation of the latent variable is $z$, whereas in this article we use $A^\text{hard}$ as the latent variable to denote the refined graph structure. In addition, we leverage the variational inference and a structure encoder $g_\phi$ to infer $A^\text{hard}$ according to label information from the E-step.

To infer the new structure $A^\text{hard}$ based on label information, we consider the ELBO of the log-likelihood function using a similar derivation developed in the conditional variational autoencoder [33] and modify the original objective of
GMNN’s M-step in (7) as follows:

\[
\log p_\phi(\hat{y}_i | \hat{Y}_{\neg i}, X) \\
\geq E_{A_{\text{hard}} \sim \mathcal{G}(A_{\text{hard}}(\phi, \hat{Y}_{\neg i}), X)} \left[ \log p_\phi(A_{\text{hard}} | \hat{Y}_{\neg i}, X) \right] \\
- KL \left( g_\phi (A_{\text{hard}} | \hat{y}_i, \hat{Y}_{\neg i}, X), p_\phi (A_{\text{hard}} | \hat{Y}_{\neg i}, X) \right).
\]

(9)

Similar to the normal ELBO, the second parameter in KL(·) is our prior distribution \( p_\phi(A_{\text{hard}} | \hat{Y}_{\neg i}, X) \) of \( A_{\text{hard}} \). To avoid ambiguity and distinguish it from the decoder, we omit its footprint \( \phi \). According to the Markov independence property that each node is conditionally independent given its neighbors, we can summarize the ELBO in (9) as follows:

\[
O_{\phi, \text{ELBO}} = E_{A_{\text{hard}} \sim \mathcal{G}(A_{\text{hard}}(\phi, \hat{Y}_{\neg i}), X)} \left[ \log p_\phi(A_{\text{hard}} | \hat{Y}, X) \right] \\
- KL \left( g_\phi (A_{\text{hard}} | \hat{y}, X), p_\phi (A_{\text{hard}} | \hat{Y}, X) \right).
\]

(10)

Note that the above ELBO can also be considered as an encoder–decoder model, i.e., the structure encoder \( g_\phi (A_{\text{hard}} | \hat{Y}, X) \) encodes the latent graph structure, and the decoder \( p_\phi (A_{\text{hard}}(\phi, \hat{Y}, X)) \) performs message passing under the better graph structure to reconstruct the labels. The implementation details of the encoder and the decoder can be found in Section IV-A.2. In the following, we describe the expected reconstruction error and the Kullback–Leibler (KL) divergence of the above ELBO.

a) Reconstruction and Reparameterization: For the expected reconstruction error in (10), the encoder \( g_\phi \) uses the labels’ information to approximate latent graph structure prior distribution \( p(A_{\text{hard}} | \hat{Y}, X) \) with variational parameters predicted by a neural network. The decoder models the likelihood \( p_\phi (A_{\text{hard}}(\phi, \hat{Y}, X)) \) to reconstruct labels with a GNN given the latent graph structure \( A_{\text{hard}} \) and node pseudo-labels’ distribution over \( \hat{Y} \).

To evaluate the expectation and make the sampling operator differentiable, we use the discrete reparameterization trick, i.e., Gumbel-Softmax [31, 32], to sample the discrete latent graph structure

\[
A_{\text{hard}} \sim \text{Gumbel}(g_\phi (A_{\text{hard}} | \hat{Y}, X)).
\]

(11)

b) KL Divergence and the Prior Over Graph: A prior distribution captures our prior belief as to which parameters would have likely generated. Generally, from the earliest probabilistic generative model of graphs developed by Erdős and Rényi [34] that assumed an independent identically probability for each possible edge, we can define the random graph priors on the edges by the independent Bernoulli distribution

\[
p(A_{\text{hard}}) = \prod_{i,j} p(A_{ij}^{\text{hard}}) = \text{Bernoulli}(\rho)
\]

(12)

where \( \rho \) is the parameter of the Bernoulli matrix distribution, and \( p(A_{ij}^{\text{hard}}) \) denotes a Bernoulli variable representing whether the edge between nodes \( v_i \) and \( v_j \) exists or not. However, as noted in Section IV-A.1a, we need to preserve the intraclass edges and eliminate the interclass edges to obtain a better graph structure for message passing. Thus, instead of making the priors of the edges on graph independent, we can consider label dependency from label similarity to make it more flexible and reasonable for message passing.

However, since the labels are discrete, label similarity for modeling label dependency is less informative, i.e., the similarity of two one-hot representation labels can only be 0 or 1. As proved in the references of label distribution [35] and knowledge distillation [36], label distribution contains more information than the one-hot label. Thereby, we also use label distribution (soft label) produced by the softmax of a network output to evaluate label similarity. Each node is represented by a \( c \)-dimensional label distribution, and we can compute the cosine similarity between the connected nodes to obtain the existence prior of each edge. Since the \( c \)-dimensional label distribution vector \( \hat{y}_i \) is nonnegative and the summation of \( \hat{y}_i \) is 1, the cosine similarity between \( \hat{y}_i \) and \( \hat{y}_j \) ranges from 0 to 1. Specifically, we define the conditional prior over graphs based on the similarity between label distributions as

\[
p(A_{ij}^{\text{hard}} | \hat{Y}, X) = \text{Bernoulli}(A_{ij} \cos(\hat{y}_i, \hat{y}_j)).
\]

(13)

We set the prior probability of the edges over the original graph positively related to the label similarity between two connected nodes. It encourages the elimination of the harmful edges of the original structure \( A \). Note that we update the prior of the latent graph structure in each EM iteration. Compared with directly using unreliable pseudo-label to remove interclass edges, the KL divergence between posterior \( g_\phi (A_{\text{hard}} | \hat{Y}, X) \) and our structure prior \( p(A_{\text{hard}} | \hat{Y}, X) \) in (10) can provide soft supervision and improve the prediction performance.

2) Decoupling Attention Architecture and Constraint: In this Section IV-A.2a, we describe the decoupling attention procedure for the implementation of (10) in the M-step. To maximize information gain during the message passing process, we should not only consider label information for structure learning but also use feature information for weight learning. Therefore, we first use hard attention \( (\text{Hard}_\phi(x)_j) \) to implement the structure encoder \( g_\phi \). Then we decompose the decoder into soft attention weight learner \( (\text{Soft}_\phi(x)_j) \) and a simple GCN for information propagation. Moreover, we impose a mutual information constraint in the attention learning process.

As shown in Fig. 3, GNN\( _p^\phi \) in the M-step consists of three parts: pair-wise hard attention, soft local attention, and a GCN

\[
\text{GNN}_{\phi}^p := \text{GCN}_{\phi} \circ \text{Soft}_\phi \circ \text{Hard}_\phi
\]

(14)

where \( \circ \) indicates the composition of operators. The running process of GNN\( _p^\phi \) in the M-step can be divided into the following three steps:

\[
A_{\text{hard}} \sim \text{Gumbel}(\text{Hard}_\phi(A, \hat{Y}_1)), \quad (\text{Encoder})
\]

\[
A_{\text{soft}} = \text{Soft}_\phi(A_{\text{hard}}, X), \quad (\text{Decoder})
\]

\[
p_\phi (\hat{Y} | A_{\text{hard}}, \hat{Y}, X) = \text{Cat}(\text{GCN}_\phi(A_{\text{soft}}, [\hat{Y} | X])), \quad (\text{Decoder})
\]

(15)

where \( A_{\text{hard}} \) and \( A_{\text{soft}} \) denote the refined graph structure with binary and soft aggregation weights, respectively. At first,
we implement the encoder \( g_\phi \) in (10) with Hard.Att\( \rho \) to encode the edges’ exist probability based on the label distribution \( \hat{Y} \) and the original structure \( A \). Then, we use the Gumbel-Softmax trick to sample a discrete refined structure \( A^{\text{hard}} \) from edges’ existing probability. Once we obtain \( A^{\text{hard}} \), we can pass it into the decoder to reconstruct labels. In the decoder, we first apply Soft.Att\( \rho \) to learn the aggregation weights \( A^{\text{soft}} \) based on features \( X \) and \( A^{\text{hard}} \). Then, a simple GCN can leverage the aggregation weights to perform message passing and reconstruct the labels.

In the sequel, we describe the details of Hard.Att\( \rho \) and Soft.Att\( \rho \). In addition, to enhance supervision and information gain during message passing, we impose a mutual information constraint.

\( a) \) **Hard Attention With Label Similarity:** To learn a refined graph structure and reduce negative disturbance, we can use label information to remove interclass edges. A natural way is to leverage ground-truth labels for labeled nodes and pseudo-labels for unlabeled nodes to identify interclass edges. However, due to the limited number of labeled nodes in the semisupervised setting for real-world datasets, directly using pseudo-labels may lead to incorrect edges’ existence because of noisy supervision signals. Aiming to model the uncertainty of edges, we apply variational inference and use the hard attention mechanism on node label distribution to obtain a better graph structure latent distribution. To further model the similarity of labels, we use the bilinear model [37] to compute the variational probability of edges between nodes \( v_i \) and \( v_j \):

\[
\text{Hard.Att} \rho (A, \hat{Y}) = A \odot \text{sigmoid}(\hat{Y}^T Q \hat{Y})
\]

where \( \odot \) denotes the element-wise product, \( Q \in \mathbb{R}^{C \times C} \) is a learnable metric matrix to measure the label similarity of \( \hat{y}_i \) and \( \hat{y}_j \), and \( C \) denotes the number of classes. Hard attention Hard.Att\( \rho (A, \hat{Y}) \) is the implementation of the encoder \( g_\phi (A^{\text{hard}} | \hat{Y}, X) \) in (10) to infer a refined graph structure based on labels’ dependency. Note that we do not need feature \( X \) as input to learn the structure since we assume that the label information is sufficient. As mentioned before, we assume that the existence probability of each edge obeys a Bernoulli distribution. Therefore, each element in the output of \( g_\phi (A^{\text{hard}} | \hat{Y}, X) \) is considered as the corresponding parameter of the edge’s existence probability distribution. Moreover, hard attention learning is also constrained by the KL divergence of our graph structure prior defined in (13), which can alleviate the lack of attention supervision and help hard attention learn a meaningful pattern.

\( b) \) **Soft Attention With Feature Dissimilarity:** With hard attention Hard.Att(\( \rho \)(\( A, \hat{Y} \)) and the refined binary graph structure \( A^{\text{hard}} \) at hand, we design soft attention according to the node features for edge weight learning. As noted in the previous section, our soft attention for message passing is encouraged to focus more on dissimilar neighbors to increase information gain, and Soft.Att\( \rho \) can be implemented as follows:

\[
h_i = s \circ \text{ReLU}(W^{\text{proj}} \cdot x_i)\\
\delta_{ij} = -\cos(h_i, h_j)\\
A^{\text{soft}}_{ij} = \frac{A^{\text{hard}}_{ij} \exp(\delta_{ij})}{\sum_{j=1}^{N} A^{\text{hard}}_{ij} \exp(\delta_{ij})}.
\]

We use each element in \( A^{\text{soft}} \) as aggregation weight to guide message passing in GCN\( \rho \). \( W^{\text{proj}} \in \mathbb{R}^{d \times m} \) is a projection head to reduce the dimension of node feature embedding \( x_i \) and \( s \in \mathbb{R}^m \) is a rescaling parameter for tuning the importance factor of each dimension. We use negative cosine similarity between the node’s hidden state to obtain the dissimilarity score of nodes and use softmax on the neighbors of each node to normalize the score to obtain the attention weights on edges. After we obtain \( A^{\text{soft}} \), \( A^{\text{soft}}_{ij} \) can be used to replace the aggregation weights \( a_{ij} \) in message passing (1) for the simple GCN\( \rho \) in the M-step. Nevertheless, other formulas to compute attention based on dissimilarity can also be used in the proposed model to implement edge weights.

\( c) \) **Neighbor Mutual Information Maximization:** In addition to learning a refined graph to reduce intraclass edges and improve positive information gain during message passing, we note that the attention weights degenerate to the average for citation networks due to the lack of supervision problem [19], [20]. As mentioned in Fig. 1, the desired aggregation weights are supposed to focus more on dissimilar intraclass neighbors. Thus, they are able to enhance nodes’ information gain during message passing. To tackle this problem, apart from the KL supervision signal for structure learning in the

![Diagram](image_url)
M-step, we impose mutual information regularization to prevent the aggregation weights from being uniform. Because mutual information of node features is intractable, we use the labels inferred by $p_\phi$ as an alternative. Denoting the neighbor label distribution by $\hat{Y}_N$, we have

$$P(\hat{Y}) = GNN^S \theta, \quad P(\hat{Y} | \hat{Y}_N) = GNN^F_{\phi}$$

$$I(\hat{Y}, \hat{Y}_N) = H(\hat{Y}) - H(\hat{Y} | \hat{Y}_N)$$

$$O_{\phi, \text{ENT}} = H(\hat{Y} | \hat{Y}_N) = \mathbb{E}_{\hat{Y}_N}[H(GNN^F_{\phi})].$$ (18)

Here, we use superscripts $P$ and $Q$ to indicate the M-step and E-step networks, respectively. In the M-step, $P(\hat{Y})$ and $P(\hat{Y} | \hat{Y}_N)$ are fixed, because they are approximated by the fixed network $GNN^S \theta$ in the E-step. $P(\hat{Y} | \hat{Y}_N)$ indicates the label distribution after label propagation by $GNN^F_{\phi}$. As mentioned before, proper aggregation weights learned by $GNN^F_{\phi}$ can help enhance information gain and reduce the uncertainty of label distribution. The mutual information $I(\hat{Y}, \hat{Y}_N)$ computes the reduction in label uncertainty after message passing in the M-step. Hence, maximizing the mutual information $I(\hat{Y}, \hat{Y}_N)$ helps learn proper aggregation weights for propagating labels. When we are optimizing the network $GNN^F_{\phi}$ in the M-step using backpropagation, there is no gradient backpropagating to $H(\hat{Y})$ given $GNN^S \theta$. Therefore, maximizing the mutual information $I(\hat{Y}, \hat{Y}_N)$ in the M-step is equivalent to minimizing the entropy (ENT) of $GNN^S \theta$. This entropy term can also be regarded as assisting in learning meaningful aggregation weights that may improve prediction confidence by integrating beneficial information from neighbors. The final objective for the M-step is

$$O_\phi = O_{\phi, \text{ELBO}} - \beta O_{\phi, \text{ENT}}$$ (19)

where the scalar $\beta \in \mathbb{R}$ is a regularization hyperparameter to weight mutual information constraint.

**B. E-Step: Stable Graph Reweighting and Feature Propagation**

To calculate the joint distribution expectation in the E-step, as shown in (8), the task of the E-step is to approximate posterior distribution of unlabeled nodes $p_\phi(Y_U | Y_L, X)$ by updating the parameters of variational $q_\phi(Y_U | X)$. Unlike GMNN [25], in which $p_\phi$ propagates labels under the original structure, we consider label dependency under a refined graph structure to reduce negative disturbance. Thus, our posterior prediction of latent labels is the marginal distribution over the latent graph $A^{\text{hard}}$ and can be computed by

$$p(Y_U | Y_L, X) = p_\phi(Y_U | \hat{Y}, X)$$

$$= \sum_{A^{\text{hard}}} p_\phi(Y_U | A^{\text{hard}}, \hat{Y}, X) p_\phi(A^{\text{hard}} | \hat{Y}, X)$$

$$\approx \frac{1}{S} \sum_{s=1}^{S} p_\phi(Y_U | A^{\text{hard}(S)}, \hat{Y}, X)$$ (20)

where $A^{\text{hard}(S)}$ is a sample from the posterior graph distribution of the hard attention encoder $g_\phi(A^{\text{hard}} | \hat{Y}, X)$ in the M-step. $S$ is the total number of samplings and is good enough to set to five in practice.

For the variational $q_\phi(Y_U | X)$, unlike GMNN [25], which propagates features on the original graph, we also perform feature propagation in the E-step according to more reasonable weights learned from the M-step. However, sampling a discrete variable from Gumbel-Softmax may cause a large variance [31]. If we directly sample the graph structure from hard attention in the M-step and pass it to the E-step, the training process may be unstable and suboptimal. To alleviate the variance, one approach is to sample $S$ times and average $A^{\text{hard}(S)}$. From the averaged $A^{\text{hard}(S)}$’s, we can obtain weights for E-step feature propagation. However, it may be expensive when $S$ is large. To solve these issues, we use a simple yet effective fusion of hard and soft attention to obtain stable weights $A^{\text{stable}}$ for message passing in the E-step

$$A^{\text{stable}} = \text{Hard}_\text{Att}_\phi(A, \hat{Y}) \odot \text{Soft}_\text{Att}_\phi(A, X).$$ (21)

The stable weights $A^{\text{stable}}$ are more reliable for message passing. Note that soft attention learns weights by node features over the graph, and hard attention probability can be regarded as a rescaling factor indicating label similarity. Hence, fused weights are more stable and representative in terms of both the features and labels. Consequently, unlike (8) in GMNN, the optimal state of our feature propagator $q_\phi(y_i | X, A^{\text{stable}})$ needs to satisfy

$$q_\phi(y_i | X, A^{\text{stable}}) \approx p_\phi(y_i | \hat{Y}, X), \quad i \in U.$$ (22)

For the unlabeled data $U$, we train $q_\phi$ by minimizing the reverse KL divergence between our feature propagator $q_\phi(y_i | X, A^{\text{stable}})$ and the fixed target posterior $p_\phi(y_i | \hat{Y}, X)$ in (20). For the labeled data, we compute the log likelihood directly. Combining these two parts of unlabeled and labeled samples, we intend to maximize the following objective function $O_\theta$:

$$O_\theta = \sum_{i \in L} \log q_\phi(y_i | X, A^{\text{stable}})$$

$$+ \lambda \sum_{i \in U} \mathbb{E}_{p_\phi(Y_U | X, X)}[\log q_\phi(y_i | X, A^{\text{stable}})]$$ (23)

where $\lambda \in \mathbb{R}$ is a hyperparameter to balance the losses of labeled and unlabeled samples.

**C. Optimization and Algorithm**

The detailed algorithm is summarized in Algorithm 1. We first train a feature propagator $q_\phi$ using labeled data with the entropy regularizer to encourage the initial pseudo-label distribution with sharper distribution, i.e., distribution with large kurtosis. Then we alternatively optimize $p_\phi$ and $q_\phi$ using the EM procedure until convergence. In the M-step, we first update the pseudo-labels by $q_\phi$, and then run the decoupling attention to obtain a better structure $A^{\text{hard}}$ and weights $A^{\text{stable}}$ for message passing. In the E-step, we first update the pseudo-labels by $p_\phi$, and then use the stable reweighting graph $A^{\text{stable}}$ for $q_\phi$ to perform message passing on features. The final result is reported using $q_\phi$. 

---

**Algorithm 1:**

1. **Initialization:**
   - Set $p_\phi$, $q_\phi$, and other parameters.
   - Sample $A^{\text{stable}}, \hat{Y}$.

2. **M-Step:**
   - **Update $A^{\text{hard}}$:**
     - **Sample $A^{\text{hard}(S)}$:**
       - $A^{\text{hard}(S)} \sim \text{Gumbel-Softmax}(A^{\text{stable}}, \beta)$
     - $p_\phi(A^{\text{hard}} | \hat{Y}, X)$
   - **Update $q_\phi$:**
     - $q_\phi(y_i | X, A^{\text{stable}}) \approx p_\phi(y_i | \hat{Y}, X)$

3. **E-Step:**
   - **Update $p_\phi$:**
     - $p_\phi(Y_U | Y_L, X)$
   - **Update $q_\phi$:**
     - $q_\phi(y_i | X, A^{\text{stable}})$

4. **Repeat:**

---

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Algorithm 1 Optimization of the Proposed GDAMN Method.
1: **Input:** Graph adjacency matrix $A$, node feature $X$, labeled nodes $Y_L$, and unlabeled nodes $Y_U$.
2: **Output:** Labels $Y_U$ for the unlabeled sets $U$.
3: Pre-train feature propagator to obtain initial $q_0$ with ENT regularizer.
4: **while** not converge do
5:  □ **M-Step:** Graph Inference and Decoupling Architecture
6:  - Annotate the unlabeled objects $\hat{Y}_U$ by $q_0$.
7:  - Calculate $g_\phi(A^{\text{hard}}|\hat{Y}, X)$ by employing Hard_Att on labels according to Eq. (16).
8:  - Sample a discrete graph structure $A^{\text{hard}} \sim g_\phi(A^{\text{hard}}|\hat{Y}, X)$ by Gumbel reparameter trick.
9:  - Calculate aggregation weights $A^{\text{soft}}$ by employing Soft_Att on features and the refined structure $A^{\text{hard}}$ according to Eq. (17).
10: - Predict all nodes’ label distribution $p_\theta(\hat{Y}|A^{\text{hard}}, \hat{Y}, X)$ according to the last equation in Eq. (15).
11: - Calculate the M-step optimization object $O_\phi$ according to Eq. (19) and update parameters $\phi$ by gradient descent.
12: □ **E-Step:** Stable Graph Re-weighting and Feature Propagation
13:  - According to Eq. (20) to infer $p_\theta$ to obtain the latent graph distribution $A^{\text{hard}}$ and the corresponding $A^{\text{soft}}$.
14:  - Calculate $A^{\text{stable}}$ according to Eq. (21).
15:  - Predict all nodes’ label distribution $q_\theta(Y|A^{\text{stable}}, X)$ according to Eq. (24).
16:  - Calculate the E-step optimization object $O_\theta$ according to Eq. (23) and update parameters $\theta$ by gradient descent.
17: **end while**
18: Predict the unlabeled objects $\hat{Y}_U$ by $q_\theta$.

In the inference time, we can use both $p_\theta$ and $q_\theta$ to infer the labels of unlabeled nodes. However, we find that $q_\theta$ consistently outperforms $p_\theta$ in practice. Thus, we use $q_\theta$ with the stable weights $A^{\text{stable}}$ learned by $p_\theta$ to infer unlabeled nodes

$$q_\theta(Y|A^{\text{stable}}, X) = \text{Cat} \left( \text{GNN}_\theta^O(A^{\text{stable}}, X) \right).$$

V. EXPERIMENTS

In this section, extensive experiments are performed to verify the effectiveness and superiority of our method. First, we report the experimental results on six ubiquitous benchmark graph datasets for node classification. Afterward, visualized results and ablation studies are discussed to show the effectiveness of learned attention.

A. Datasets

We first introduce the graph datasets used for node classification. Similar to previous studies [8], [9], [38], [39], we use six benchmark datasets from [39]–[41] for performance evaluation: three basic citation datasets: Cora, Citeseer, and Pubmed, two coauthor datasets: Coauthor-context-surrounding (CS) and Coauthor-Phy, and one Open Graph Benchmark (OGB) dataset: Arxiv. The detailed statistics of these six datasets are presented in Table II.

| Dataset       | Nodes | Edges | Features | Classes | Train/Val/Test |
|---------------|-------|-------|----------|---------|----------------|
| Cora          | 2,708 | 5,278 | 1,433    | 7       | 140/500/1,000  |
| Citeseer      | 3,327 | 4,552 | 3,703    | 6       | 120/500/1,000  |
| Pubmed        | 19,717| 44,324| 500      | 3       | 60/500/1,000   |
| Coauthor-CS   | 18,333| 81,894| 6,805    | 15      | 300/450/17,583 |
| Coauthor-Phy  | 34,493| 247,962| 8,415    | 5       | 100/150/34,243 |
| OGB-Arxiv     | 169,343| 1,166,243| 128      | 40      | 90,941/29,799/48,603 |

The Arxiv dataset is from the OGB [41], where nodes represent computer science papers on arXiv. Papers are classified into 40 classes based on the arXiv subject area. Unlike the basic citation datasets that use bag-of-words as node features, the node features in Arxiv are computed as the average word embedding of all words in the paper.

We use the same data partition and preprocessing as in [17], [38], and [41] for citation datasets and [39] for coauthor datasets. The evaluation metric is the prediction accuracy of the test nodes.

B. Baselines

We compare our method with three non-GNN models and GNN-based methods. The baseline models are logistic regression (LogReg), multilayer perceptron (MLP), and label propagation (LPA) [21]. The first two methods are attribute-based models that only use the feature representation of each node. The label of each node indicates the most active field of the author.

The Arxiv dataset is from the OGB [41], where nodes represent computer science papers on arXiv. Papers are classified into 40 classes based on the arXiv subject area. Unlike the basic citation datasets that use bag-of-words as node features, the node features in Arxiv are computed as the average word embedding of all words in the paper.

We use the same data partition and preprocessing as in [17], [38], and [41] for citation datasets and [39] for coauthor datasets. The evaluation metric is the prediction accuracy of the test nodes.

1) Classical GNNs

a) GCN [8]: GCN is the pioneer to perform linear approximation to spectral graph convolutions.

b) SGC [42]: Simplifying graph convolution (SGC) reduces GCNs’ complexity by removing nonlinearities and collapsing weight matrices between consecutive layers. This method can also be regarded as decoupling the feature transformation and propagation.
c) APPNP [43]: Approximate personalized propagation of neural prediction (APPNP) combines GNN with personalized PageRank to separate the neural network from the propagation scheme.
d) GraphSAGE [26]: GraphSAGE learns a function that generates embeddings by sampling and aggregating features from a node’s local neighborhood.
e) GAT [9]: GAT is a GNN that applies the attention mechanism to node features to learn edge weights.

2) Label Information GNNs

a) CS-GNN [14]: The CS-GNN uses feature and label smoothness to help the attention mechanism selectively aggregate neighborhood information and reduce negative disturbances.
b) ALaGCN and ALaGAT [17]: ALaGCN and ALaGAT propose a novel metric and integrate them into an adaptive-layer module to make individual decisions at each round of neighborhood aggregation. They estimate whether neighborhood aggregation is harmful or unnecessary by calculating the metric based on label information.
c) GMNN [25]: GMNN is also an EM-based method that models the joint distribution of nodes and labels with a conditional random field and uses two GCNs to propagate features and labels under the original structure.
d) GCN-LPA [16]: GCN-LPA can also be seen as learning attention weights based on training set labels. However, the input of attention retains node features, and it uses LPA as regularization to assist in learning proper edge weights.
e) GCN-C&S [23]: The GCN-C&S combines label propagation with a basic predictor to learn the residual. For a fair comparison, we choose the GCN as the basic model since our GDAMN also uses the GCN as a predictor. For the scaling strategy of C&S, we choose the Autoscale that works more reliably than FDiff-scale, which is also mentioned from their official code.

d) Deep GNNs

a) DeeperGCN [44]: It combines residual/dense connections and dilated convolutions and adapts them to GCN architectures to successfully train very deep GCNs.
b) DAGNN [45]: The deep adaptive GNN (DAGNN) decouples feature transformation and propagation to learn node representation from larger receptive fields. Moreover, it proposes an adaptive adjustment to balance the information from local and global neighbors for each node.

For ease of comparison, we use the main reported results from [17], [25], [39]. Moreover, for the missing results of most baselines on the coauthor and arXiv datasets, we rerun their official codes over 20 runs. Note that for the GCN-LPA and GCN-C&S, the original articles use the fully supervised data splits which use the train/val/test as 60%/20%/20% for the citation and coauthor datasets. For a fair comparison, we also rerun their codes under our semisupervised data splits in which only 20 labeled samples per class are used for training.

C. Experiment Settings

We use two hidden layers, and the dimension of the hidden layers is set as 16 for citation datasets and 64 for coauthor datasets, which are the same as in [25] and [39]. We initialize weights according to Kaiming initialization [46] and train our model for two EM iterations with 200 epochs per iteration using Adam [47]. We report the mean test accuracy when the validation accuracy is maximized over 20 experiments with different random seeds. During training, we set the initial learning rate as 0.05 and weight decay as 0.0005/0.0001 for the (citation/coauthor) datasets, and after the first EM iteration the weight decay is set to 0.0001 for the citation and coauthor datasets. For the Arxiv dataset, we set the hidden layers to three, the hidden dimension to 128, the learning rate to 0.02, the weight decays to 0, and the number of epochs to 1000 per EM iteration. We also apply dropout [48] with the dropout rate $d = 0.5$ on hidden layers and $d = 0.2$ for the attention weights. For ENT regularization, we select the parameter $\beta \in \{0.2, 0.4, 0.6, 0.8, 1\}$ using the validation set. We set the weighted parameter $\lambda$ of the cross-entropy loss on the unlabeled data part to 0.8 for all the datasets.

D. Results and Analysis

1) Performance Evaluation: The performance on node prediction accuracy is summarized in Tables III and IV. We first note that almost all the GNNs achieve much better prediction accuracy than the non-GNN models (LogReg, MLP, and LPA). Compared with the GNN-based method that combines the graph structure and feature representation of nodes, the non-GNN models merely use partial information for node prediction. In addition, our GDAMN performs better than Laplacian-based GNNs (GCN, SGC, and APPNP). These methods propagate features under the original graph structure and may suffer from negative disturbances due to in-class edges. Moreover, our GDAMN outperforms attention-based GNNs such as CS-GNN, GAT, and ALaGAT because our decoupling attention can learn more meaningful attention weights from both the features and labels. Compared with the label propagation methods (GCN-LPA, GMNN, and GCN-C&S), our model also achieves better performance in terms of higher prediction accuracies on most datasets. Although our performance is slightly worse for the Pumbed dataset than that of the GMNN method, our prediction accuracy is still better than other GNNs. The reason might be that there are fewer negative disturbances and label dependencies in the Pubmed dataset since it has only three classes.

2) Ablation Study: In this section, we compare GDAMN with its four variants on all the datasets to validate the effectiveness of each component.

1) GMNN (with GAT): A natural variant that we replace the GCN with GAT in GMNN to apply the attention mechanism on node labels.
2) GDAMN (w/o ENT): GDAMN without mutual information constraint.
3) **GDAMN (w/o Hard):** GDAMN without hard attention and variational inference for structure learning in the M-step.

4) **GDAMN (w/o Soft):** GDAMN without soft attention for weight learning in the M-step.

As shown in Table III, the performance of each variant degrades for all the datasets, which demonstrates the effect of the ENT regularizer and decoupling two-step attention design. The GMNN (with GAT) is slightly better than the GMNN. It has been shown that performing attention on labels is helpful for message passing. Performance degeneration of GDAMN (w/o Hard) and GDAMN (w/o Soft) confirms...
that both the graph structure and weights are important for message passing in node classification tasks. Apart from them, GDAMN (w/o ENT) outperforms GMNN (with GAT) which shows that our decoupling attention design is more useful for each node to obtain information gain during message passing.

3) Effectiveness of Stable Graph Reweighting: As discussed in Section IV-B, when reweighting the graph of the E-step, training of the GCN may be unstable due to the variance of the sample discrete graph structure from the Gumbel-softmax. To show the superiority of our stable reweighting in the E-step, we tested the node classification performance under different graph sampling numbers \( S \) and varied it from 1 to 10 on the Cora and Citeseer datasets, and 0 indicates our stable reweighting. The results are shown in Fig. 4, and when increasing the sampling number \( S \), the performance increases slowly. However, the performance of stable reweighting is consistently higher than direct sampling of different \( S \), which shows that the stable weights \( A_{\text{stable}} \) in (21) are more efficient and stable for message passing.

4) Learned Attention: We then visualize the attention to demonstrate the effectiveness of our learning procedure. As stated in Section I, we expect the intraclass connectivities to be more potent than the interclass connectivities. We calculate the mean connectivity strengths of intra- and interclasses of the Cora dataset (high connectivity strengths mean more attention weights are concentrated on the edge between the two classes) and visualize the matrix in Fig. 5. Because GMNN performs message passing on the original graph, its connectivity strength is the same as that of the GCN. Meanwhile, the result of feature-based attention model GAT is similar to that of GCN (GMNN), since it omits label dependency and lacks supervision. However, if we replace the GCN in GMNN with GAT and apply attention to labels, the results become better. It shows the importance of label information for attention learning. Compared with the GAT and GMNN methods, our GDAMN method obtains higher scores for the diagonal part, which implies higher intraclass strength. Moreover, the information ENT regularizer and hard attention constrained by the KL divergence between our graph prior can provide more supervision and significantly help reduce the interclass weights ratio.

5) Retrain With Vanilla GCN: To further verify the effectiveness of the learned aggregation weights, we compare the training loss and test accuracy using a vanilla GCN with different edge weights. The weights are the graph Laplacian computed by the initial adjacent matrix, the oracle graph Laplacian computed by removing all the interclass edges, and the learned aggregation weights computed by the proposed method.
GDAMN. As shown in Fig. 6, in the training stage, the GCN with the learned attention weights learned by our GDAMN achieves a similar convergence rate to the oracle and significantly faster than GCN with the original adjacent matrix. In the test stage, our performance also significantly outperforms the GCN with the original adjacent matrix. In this sense, our method can also be used as a graph learner to refine the edge strength of the aggregation weights, which is compatible with any GNN with more sophisticated structures.

6) Analysis of Time Complexity: For standard GNNs, the time complexity of a single GCN layer is $O(|V|F'F + |E|F')$, where $F$ and $F'$ are the numbers of input and hidden dimensions, respectively, and $|V|$ and $|E|$ are the numbers of nodes and edges, respectively. $O(|V|F')$ represents the time complexity of the feature transform for each node, and $O(|E|F')$ represents the time complexity of message passing along edges. For GAT, since it uses additive attention, the time complexity is also $O(|V|F'F'' + |E|F'')$, which is on par with GCN.

For EM models, GMNN uses the two GCNs, so the time complexity is also $O(|V|F'F'' + |E|F'')$. For GDAMN, the time complexity of a single GNN$^P$ layer is $O(|V|F'F + |E|CC + |E_F|F''')$, where $F'''$ is the number of projection feature dimensions in soft attention. $C$ is the class numbers, and $|E_F|$ is the number of latent structure edges. Specifically, the cost for bilinear hard attention is $O(|E|CC)$, and the cost for soft attention and message passing is $O(|E_F|F''')$. Since we need two-step attention computation, this complexity is slightly higher than the baseline methods such as GAT and GMNN. In our experiments, we empirically noted that GDAMN converges in three EM iterations.

VI. CONCLUSION

In this article, we have proposed a GDAMN to decouple the attention learning procedure on the graph into hard and soft attention. Hard attention learns the graph structure based on labels, and soft attention learns edge weights based on features. To avoid the influence of insufficient label information in the semisupervised node classification scenario, we use the EM framework to approximate label distribution and apply variational inference to model the uncertainty of the structure in the M-step. Moreover, we impose a novel structure prior and the neighbor mutual information constraint in the learning process to obtain better hard and soft attention. The experiments on five benchmark datasets demonstrate that our model outperforms the state-of-the-art methods with more reasonable aggregation weights for both feature and label propagation.

Our work can also be regarded as a general framework that considers the features and labels to learn the graph structure and edge weights. In this work, we use the simple GCN as a propagator for the feature and label. One of our future works is to combine GDAMN with more complex GNNs and apply it to computer vision or natural language processing. It is also promising to explore other graph structure priors that are reasonable for graph-level and edge-level tasks. These priors can extend our work to graph-level and edge-level tasks in the future. Furthermore, it is interesting to investigate how to combine effective sampling techniques with our methods to learn subgraph structures to deal with extremely large graphs.

REFERENCES

[1] J. Qiu, J. Tang, H. Ma, Y. Dong, K. Wang, and J. Tang, “DeepInf: Modeling influence locality in large social networks,” in Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2018, pp. 2110–2119.
[2] W. Liang and W. Zhang, “Learning social relations and spatiotemporal trajectories for next check-in inference,” IEEE Trans. Neural Netw. Learn. Syst., early access, Oct. 20, 2020, doi: 10.1109/TNNLS.2020.3016737.
[3] Z. Li, H. Liu, Z. Zhang, T. Liu, and N. N. Xiong, “Learning knowledge graph embedding with heterogeneous relation attention networks,” IEEE Trans. Neural Netw. Learn. Syst., early access, Feb. 19, 2021, doi: 10.1109/TNNLS.2021.3055147.
[4] N. Park, A. Kan, X. L. Dong, T. Zhao, and C. Faloutsos, “Estimating node importance in knowledge graphs using graph neural networks,” in Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Jul. 2019, pp. 596–606.
[5] K. Do, T. Tran, and S. Venkatesh, “Graph transformation policy network for chemical reaction prediction,” in Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Jul. 2019, pp. 750–760.
[6] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, “Graph convolutional neural networks for web-scale recommender systems,” in Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Jul. 2018, pp. 974–983.
Mingyuan Bai received the B.Com. degree (Hons.) in business analytics degree from The University of Sydney, Darlington, NSW, Australia, in 2017. She is currently pursuing the Ph.D. degree in business analytics, commerce with The University of Sydney Business School, The University of Sydney.

Her main research interests are in machine learning, data analytics, and tensor learning.

Jian Pu (Member, IEEE) received the Ph.D. degree from Fudan University, Shanghai, China, in 2014. He was an Associate Professor with the School of Computer Science and Software Engineering, East China Normal University, Shanghai, from 2016 to 2019, and a Postdoctoral Researcher of the Institute of Neuroscience, Chinese Academy of Sciences, Beijing, China, from 2014 to 2016. He is currently a Young Principal Investigator with the Institute of Science and Technology for Brain-Inspired Intelligence (ISTBI), Fudan University. His current research interests include machine learning, computer vision, and medical image analysis.

Junping Zhang (Senior Member, IEEE) received the B.S. degree in automation from Xiangtan University, Xiangtan, China, in 1992, the M.S. degree in control theory and control engineering from Hunan University, Changsha, China, in 2000, and the Ph.D. degree in intelligent systems and pattern recognition from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2003. He has been a Professor with the School of Computer Science, Fudan University, Shanghai, China, since 2006. His research interests include machine learning, image processing, biometric authentication, and intelligent transportation systems.

Dr. Zhang has been an Associate Editor of the IEEE Intelligent Systems since 2009 and was an Associate Editor of the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS (2010–2018).

Junbin Gao received the B.Sc. degree in computational mathematics from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 1982, and the Ph.D. degree from the Dalian University of Technology, Dalian, China, in 1991. He was a Professor in computer science with the School of Computing and Mathematics, Charles Sturt University, Bathurst, NSW, Australia, from 1982 to 2001, he was an Associate Lecturer, a Lecturer, an Associate Professor, and a Professor of mathematics with HUST. He was a Senior Lecturer and a Lecturer in computer science with the University of New England, Armidale, NSW, Australia, from 2001 to 2005. He is a Professor of big data analytics with The University of Sydney Business School, The University of Sydney. His main research interests include machine learning, data analytics, Bayesian learning and inference, and image analysis.