
GMNN: Graph Markov Neural Networks

Meng Qu^{1 2} Yoshua Bengio^{1 2 3} Jian Tang^{1 3 4}

Abstract

This paper studies semi-supervised object classification in relational data, which is a fundamental problem in relational data modeling. The problem has been extensively studied in the literature of both statistical relational learning (e.g. relational Markov networks) and graph neural networks (e.g. graph convolutional networks). Statistical relational learning methods can effectively model the dependency of object labels through conditional random fields for collective classification, whereas graph neural networks learn effective object representations for classification through end-to-end training. In this paper, we propose the Graph Markov Neural Network (GMNN) that combines the advantages of both worlds. A GMNN models the joint distribution of object labels with a conditional random field, which can be effectively trained with the variational EM algorithm. In the E-step, one graph neural network learns effective object representations for approximating the posterior distributions of object labels. In the M-step, another graph neural network is used to model the local label dependency. Experiments on object classification, link classification, and unsupervised node representation learning show that GMNN achieves state-of-the-art results.

1. Introduction

We live in an interconnected world, where entities are connected through various relations. For example, web pages are linked by hyperlinks; social media users are connected through friendship relations. Modeling such relational data is an important topic in machine learning, covering a variety of applications such as entity classification (Perozzi et al., 2014), link prediction (Taskar et al., 2004) and link

classification (Dettmers et al., 2018).

Many of these applications can be boiled down to the fundamental problem of semi-supervised object classification (Taskar et al., 2007). Specifically, objects¹ are interconnected and associated with some attributes. Given the labels of a few objects, the goal is to infer the labels of other objects. This problem has been extensively studied in the literature of statistical relational learning (SRL), which develops statistical methods to model relational data. Some representative methods include relational Markov networks (RMN) (Taskar et al., 2002) and Markov logic networks (MLN) (Richardson & Domingos, 2006). Generally, these methods model the dependency of object labels using conditional random fields (Lafferty et al., 2001). Because of their effectiveness for modeling label dependencies, these methods achieve compelling results on semi-supervised object classification. However, several limitations still remain. (1) These methods typically define potential functions in conditional random fields as linear combinations of some hand-crafted feature functions, which are quite heuristic. Moreover, the capacity of such models is usually insufficient. (2) Due to the complexity of relational structures between objects, inferring the posterior distributions of object labels for unlabeled objects remains a challenging problem.

Another line of research is based on the recent progress of graph neural networks (Kipf & Welling, 2017; Hamilton et al., 2017; Gilmer et al., 2017; Veličković et al., 2018). Graph neural networks approach object classification by learning effective object representations with non-linear neural architectures, and the whole framework can be trained in an end-to-end fashion. For example, the graph convolutional network (GCN) (Kipf & Welling, 2017) iteratively updates the representation of each object by combining its own representation and the representations of the surrounding objects. These approaches have been shown to achieve state-of-the-art performance because of their effectiveness in learning object representations on relational data. However, one critical limitation is that the labels of objects are independently predicted based on their representations. In other words, the joint dependency of object labels is ignored.

¹In this paper, we will use “object” and “node” interchangeably to refer to entities in graphs, because they are different terminologies used in the literature of statistical relational learning and graph neural networks.

¹Mila - Québec AI Institute ²University of Montréal ³Canadian Institute for Advanced Research (CIFAR) ⁴HEC Montréal. Correspondence to: Meng Qu <meng.qu@umontreal.ca>, Jian Tang <jian.tang@hec.ca>.

In this paper, we propose a new approach called the Graph Markov Neural Network (GMNN), which combines the advantages of both statistical relational learning and graph neural networks. A GMNN is able to learn effective object representations as well as model label dependency between different objects. Similar to SRL methods, a GMNN includes a conditional random field (Lafferty et al., 2001) to model the joint distribution of object labels conditioned on object attributes. This framework can be effectively and efficiently optimized with the variational EM framework (Neal & Hinton, 1998), alternating between an inference procedure (E-step) and a learning procedure (M-step). In the learning procedure, instead of maximizing the likelihood function, the training procedure for GMNNs optimizes the pseudolikelihood function (Besag, 1975) and parameterizes the local conditional distributions of object labels with a graph neural network. Such a graph neural network can well model the dependency of object labels, and no hand-crafted potential functions are required. For inference, since exact inference is intractable, we use a mean-field approximation (Oppor & Saad, 2001). Inspired by the idea of amortized inference (Gershman & Goodman, 2014; Kingma & Welling, 2014), we further parameterize the posterior distributions of object labels with another graph neural network, which is able to learn useful object representations for predicting object labels. With a graph neural network for inference, the number of parameters can be significantly reduced, and the statistical evidence can be shared across different objects in inference (Kingma & Welling, 2014).

Our GMNN approach is very general. Though it is designed for object classification, it can be naturally applied to many other applications, such as unsupervised node representation learning and link classification. Experiment results show that GMNNs achieve state-of-the-art results on object classification and unsupervised node representation learning, as well as very competitive results on link classification.

2. Related Work

Statistical Relational Learning. In the literature of statistical relational learning (SRL), a variety of methods have been proposed for semi-supervised object classification. The basic idea is to model label dependency with probabilistic graphical models. Many early methods (Koller & Pfeffer, 1998; Friedman et al., 1999; Getoor et al., 2001a;b; Xiang & Neville, 2008) are built on top of directed graphical models. However, these methods can only handle acyclic dependencies among objects, and their performance for prediction is usually limited. Due to such weaknesses, many later SRL methods employ Markov networks (e.g., conditional random fields (Lafferty et al., 2001)), and representative methods include relational Markov networks (RMN) (Taskar et al., 2002) and Markov logic networks (MLN) (Richardson &

Domingos, 2006; Singla & Domingos, 2005). Though these methods are quite effective, they still suffer from several challenges. (1) Some hand-crafted feature functions are required for specifying the potential function, and the whole framework is designed as a log-linear model by combining different feature functions, so the capacity of such frameworks is quite limited. (2) Inference remains challenging due to the complicated relational structures among objects. Our proposed GMNN method overcomes the above challenges by using two different graph neural networks, one for modeling the label dependency and another for approximating the posterior label distributions, and the approach can be effectively trained with the variational EM algorithm.

Graph-based Semi-supervised Classification. Another category of related work is graph-based semi-supervised classification. For example, the label propagation methods (Zhu et al., 2003; Zhou et al., 2004) iteratively propagate the label of each object to its neighbors. However, these methods can only model the linear dependency of object labels, while in our approach a non-linear graph neural network is used to model label dependency, which has greater expressive power. Moreover, GMNNs can also learn useful object representations for predicting object labels.

Graph Neural Networks. Another closely related research area is that of graph neural networks (Dai et al., 2016; Kipf & Welling, 2017; Hamilton et al., 2017; Gilmer et al., 2017; Veličković et al., 2018), which can learn useful object representations for predicting object labels. Essentially, the object representations are learned by encoding local graph structures and object attributes, and the whole framework can be trained in an end-to-end fashion. Because of their effectiveness in learning object representations, they achieve state-of-the-art results in object classification. However, existing methods usually ignore the dependency between object labels. With GMNNs, besides learning object representations, we also model the joint dependency of object labels by introducing conditional random fields.

GNN for PGM Inference. There are also some recent studies (Yoon et al., 2018) using graph neural networks for inference in probabilistic graphical models. Compared with their work, our work focuses on statistical relational learning while their work puts more emphasis on standard graphical models. Moreover, our work utilizes two graph neural networks for both inference and learning, while their work only uses one graph neural network for inference.

3. Problem Definition & Preliminary

3.1. Problem Definition

Relational data, in which objects are interconnected via different relations, are ubiquitous in the real world. Modeling

relational data is an important direction in machine learning with various applications, such as object classification and link prediction. In this paper, we focus on a fundamental problem, semi-supervised object classification, as many other applications can be reformulated into this problem.

Formally, the problem of semi-supervised object classification considers a graph $G = (V, E, \mathbf{x}_V)$, in which V is a set of objects, E is a set of edges between objects, and \mathbf{x}_V stands for the attributes of all the objects. The edges in E may have multiple types, which represent different relations among objects. In this paper, for simplicity, we assume all edges belong to the same type. Given the labels \mathbf{y}_L of a few labeled objects $L \subset V$, the goal is to predict the labels \mathbf{y}_U for the remaining unlabeled objects $U = V \setminus L$.

This problem has been extensively studied in the literature of both statistical relation learning (SRL) and graph neural networks (GNN). Essentially, both types of methods aim to model the distribution of object labels conditioned on the object attributes and the graph structure, i.e. $p(\mathbf{y}_V | \mathbf{x}_V, E)$. Next, we introduce the general idea of both methods. For notation simplicity, we omit E in the following formulas.

3.2. Statistical Relational Learning

Most SRL methods model $p(\mathbf{y}_V | \mathbf{x}_V)$ with conditional random fields, which employ the following formulation:

$$p(\mathbf{y}_V | \mathbf{x}_V) = \frac{1}{Z(\mathbf{x}_V)} \prod_{(n_i, n_j) \in E} \psi_{i,j}(\mathbf{y}_{n_i}, \mathbf{y}_{n_j}, \mathbf{x}_V). \quad (1)$$

Here, (n_i, n_j) is an edge in graph G , and $\psi_{i,j}(\mathbf{y}_{n_i}, \mathbf{y}_{n_j}, \mathbf{x}_V)$ is the potential score defined on the edge. Typically, the potential score is computed as a linear combination of some hand-crafted feature functions, such as logical formulae.

With this formulation, predicting the labels for unlabeled objects becomes an inference problem, i.e., inferring the posterior label distribution of the unlabeled objects $p(\mathbf{y}_U | \mathbf{y}_L, \mathbf{x}_V)$. Exact inference is usually infeasible due to the complicated structures between object labels. Therefore, some approximation inference methods are often utilized, such as loopy belief propagation (Murphy et al., 1999).

3.3. Graph Neural Network

Different from SRL methods, GNN methods simply ignore the dependency of object labels and they focus on learning effective object representations for label prediction. Specifically, the joint distribution of labels is fully factorized as:

$$p(\mathbf{y}_V | \mathbf{x}_V) = \prod_{n \in V} p(\mathbf{y}_n | \mathbf{x}_V). \quad (2)$$

Based on the formulation, GNNs will infer the label distribution $p(\mathbf{y}_n | \mathbf{x}_V)$ for each object n independently. For each

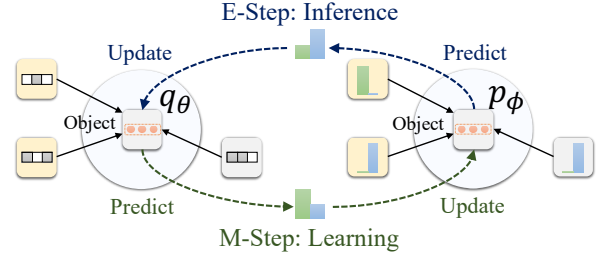


Figure 1. Framework overview. Yellow and grey squares are labeled and unlabeled objects. Grey/white grids are attributes. Histograms are label distributions of objects. Orange triple circles are object representations. GMNN is trained by alternating between an E-step and an M-step. See Sec. 4.4 for the detailed explanation.

object n , GNNs predict the label in the following way:

$$\mathbf{h} = g(\mathbf{x}_V, E) \quad p(\mathbf{y}_n | \mathbf{x}_V) = \text{Cat}(\mathbf{y}_n | \text{softmax}(W\mathbf{h}_n)),$$

where $\mathbf{h} \in \mathbb{R}^{|V| \times d}$ is the representations of all the objects, and $\mathbf{h}_n \in \mathbb{R}^d$ is the representation of object n . $W \in \mathbb{R}^{K \times d}$ is a linear transformation matrix, with d as the representation dimension and K as the number of label classes. Cat stands for categorical distributions. Basically, GNNs focus on learning a useful representation \mathbf{h}_n for each object n . Specifically, each \mathbf{h}_n is initialized as the attribute representation of object n . Then each \mathbf{h}_n is iteratively updated according to its current value and the representations of n 's neighbors, i.e. $\mathbf{h}_{\text{NB}(n)}$. For the updating function, the graph convolutional layer (GC) (Kipf & Welling, 2017) and the graph attention layer (GAT) (Veličković et al., 2018) can be used, or in general the neural message passing layer (Gilmer et al., 2017) can be utilized. After multiple layers of update, the final object representations are fed into a linear softmax classifier for label prediction. The whole framework can be trained in an end-to-end fashion with a few labeled objects.

4. GMNN: Graph Markov Neural Network

In this section, we introduce our approach called the Graph Markov Neural Network (GMNN) for semi-supervised object classification. The goal of GMNN is to combine the advantages of both the statistical relational learning methods and graph neural networks, such that we can learn useful objective representations for predicting object labels, as well as model the dependency between object labels. Specifically, GMNN models the joint distribution of object labels conditioned on object attributes, i.e. $p(\mathbf{y}_V | \mathbf{x}_V)$, by using a conditional random field, which is optimized with a pseudo-likelihood variational EM framework. In the E-step, a graph neural network is used to learn object representations for label prediction. In the M-step, another graph neural network is employed to model the local dependency of object labels. Next, we introduce the details of the GMNN approach.

4.1. Pseudolikelihood Variational EM

Following existing SRL methods, we use a conditional random field as in Eq. (1) to model the joint distribution of object labels conditioned on object attributes, i.e. $p_\phi(\mathbf{y}_V|\mathbf{x}_V)$, where the potential is defined over each edge, and ϕ is the model parameters. For now, we ignore the specific formulation of the potential function, and we will discuss it later.

We learn the model parameters ϕ by maximizing the log-likelihood function of the observed object labels, i.e. $\log p_\phi(\mathbf{y}_L|\mathbf{x}_V)$. However, directly maximizing the log-likelihood function is difficult, since many object labels are unobserved. Therefore, we instead optimize the evidence lower bound (ELBO) of the log-likelihood function:

$$\log p_\phi(\mathbf{y}_L|\mathbf{x}_V) \geq \mathbb{E}_{q_\theta(\mathbf{y}_U|\mathbf{x}_V)}[\log p_\phi(\mathbf{y}_L, \mathbf{y}_U|\mathbf{x}_V) - \log q_\theta(\mathbf{y}_U|\mathbf{x}_V)], \quad (3)$$

where $q_\theta(\mathbf{y}_U|\mathbf{x}_V)$ can be any distributions over \mathbf{y}_U , and the equation holds when $q_\theta(\mathbf{y}_U|\mathbf{x}_V) = p_\phi(\mathbf{y}_U|\mathbf{y}_L, \mathbf{x}_V)$. According to the variational EM algorithm (Neal & Hinton, 1998), such a lower bound can be optimized by alternating between a variational E-step and an M-step. In the variational E-step (a.k.a., inference procedure), the goal is to fix p_ϕ and update the variational distribution $q_\theta(\mathbf{y}_U|\mathbf{x}_V)$ to approximate the true posterior distribution $p_\phi(\mathbf{y}_U|\mathbf{y}_L, \mathbf{x}_V)$.

In the M-step (a.k.a., learning procedure), we fix q_θ and update p_ϕ to maximize the likelihood function below:

$$\ell(\phi) = \mathbb{E}_{q_\theta(\mathbf{y}_U|\mathbf{x}_V)}[\log p_\phi(\mathbf{y}_L, \mathbf{y}_U|\mathbf{x}_V)]. \quad (4)$$

However, directly optimizing the likelihood function can be difficult, as we have to deal with the partition function in p_ϕ . To avoid computing the partition function, we instead optimize the pseudolikelihood function (Besag, 1975) below:

$$\begin{aligned} \ell_{PL}(\phi) &\triangleq \mathbb{E}_{q_\theta(\mathbf{y}_U|\mathbf{x}_V)}\left[\sum_{n \in V} \log p_\phi(\mathbf{y}_n|\mathbf{y}_{V \setminus n}, \mathbf{x}_V)\right] \\ &= \mathbb{E}_{q_\theta(\mathbf{y}_U|\mathbf{x}_V)}\left[\sum_{n \in V} \log p_\phi(\mathbf{y}_n|\mathbf{y}_{\text{NB}(n)}, \mathbf{x}_V)\right], \end{aligned} \quad (5)$$

where $\text{NB}(n)$ is the neighbor set of n , and the equation is based on the independence properties of $p_\phi(\mathbf{y}_V|\mathbf{x}_V)$ derived from its formulation, i.e. Eq. (1). The pseudolikelihood approach is widely used for learning Markov networks (Kok & Domingos, 2005; Richardson & Domingos, 2006). Next, we introduce the details of the inference and learning steps.

4.2. Inference

The inference step aims to compute the posterior distribution $p_\phi(\mathbf{y}_U|\mathbf{y}_L, \mathbf{x}_V)$. Due to the complicated relational structures between object labels, exact inference is intractable.

Therefore, we approximate it with another variational distribution $q_\theta(\mathbf{y}_U|\mathbf{x}_V)$. Specifically, we use the mean-field method (Oppen & Saad, 2001), in which q_θ is formulated as:

$$q_\theta(\mathbf{y}_U|\mathbf{x}_V) = \prod_{n \in U} q_\theta(\mathbf{y}_n|\mathbf{x}_V). \quad (6)$$

Here, n is the index of unlabeled objects. In the variational distribution, all object labels are assumed to be independent.

To model the distribution of each object label in q_θ , we follow the idea of amortized inference (Gershman & Goodman, 2014; Kingma & Welling, 2014), and parameterize $q_\theta(\mathbf{y}_n|\mathbf{x}_V)$ with a graph neural network (GNN), which learns effective object representations for label prediction:

$$q_\theta(\mathbf{y}_n|\mathbf{x}_V) = \text{Cat}(\mathbf{y}_n|\text{softmax}(W_\theta \mathbf{h}_{\theta,n})). \quad (7)$$

Specifically, $q_\theta(\mathbf{y}_n|\mathbf{x}_V)$ is formulated as a categorical distribution, and the probability of each class is calculated by a softmax classifier based on the object representation $\mathbf{h}_{\theta,n}$. The representation $\mathbf{h}_{\theta,n}$ is learned by a GNN model with the object attributes \mathbf{x}_V as features, and θ as parameters. We denote the GNN model as GNN_θ . With GNN_θ , we can improve inference by learning useful representations of objects from their attributes and local connections. Besides, by sharing GNN_θ across different objects, we can significantly reduce the number of parameters required for inference, which is more efficient (Kingma & Welling, 2014).

With the above mean-field formulation, if we fix distribution $q_\theta(\mathbf{y}_{\text{NB}(n) \cap U}|\mathbf{x}_V)$, then the optimum of $q_\theta(\mathbf{y}_n|\mathbf{x}_V)$, denoted by $q^*(\mathbf{y}_n|\mathbf{x}_V)$, is specified by the following condition:

$$\begin{aligned} \log q^*(\mathbf{y}_n|\mathbf{x}_V) &= \\ &\mathbb{E}_{q_\theta(\mathbf{y}_{\text{NB}(n) \cap U}|\mathbf{x}_V)}[\log p_\phi(\mathbf{y}_n|\mathbf{y}_{\text{NB}(n)}, \mathbf{x}_V)] + \text{const}. \end{aligned} \quad (8)$$

See appendix for the proof. Bases on that, for each object n , we optimize $q_\theta(\mathbf{y}_n|\mathbf{x}_V)$ with a method similar to Salakhutdinov & Larochelle (2010). More specifically, we start by using $q_\theta(\mathbf{y}_{\text{NB}(n) \cap U}|\mathbf{x}_V)$ to compute $q^*(\mathbf{y}_n|\mathbf{x}_V)$, which is further treated as target to update $q_\theta(\mathbf{y}_n|\mathbf{x}_V)$. Computing $q^*(\mathbf{y}_n|\mathbf{x}_V)$ in Eq. (8) relies on computing the expectation with respect to $q_\theta(\mathbf{y}_{\text{NB}(n) \cap U}|\mathbf{x}_V)$. We estimate the expectation by drawing a sample from $q_\theta(\mathbf{y}_{\text{NB}(n) \cap U}|\mathbf{x}_V)$, yielding:

$$\begin{aligned} &\mathbb{E}_{q_\theta(\mathbf{y}_{\text{NB}(n) \cap U}|\mathbf{x}_V)}[\log p_\phi(\mathbf{y}_n|\mathbf{y}_{\text{NB}(n)}, \mathbf{x}_V)] \\ &\simeq \log p_\phi(\mathbf{y}_n|\hat{\mathbf{y}}_{\text{NB}(n)}, \mathbf{x}_V). \end{aligned} \quad (9)$$

In the above formula, $\hat{\mathbf{y}}_{\text{NB}(n)} = \{\hat{\mathbf{y}}_{n'}\}_{n' \in \text{NB}(n)}$ is defined as below. For each unlabeled neighbor n' of object n , we sample $\hat{\mathbf{y}}_{n'} \sim q_\theta(\mathbf{y}_{n'}|\mathbf{x}_V)$, and for each labeled neighbor n' of object n , $\hat{\mathbf{y}}_{n'}$ is set as the ground-truth label. In practice, we find that using one sample from $q_\theta(\mathbf{y}_{\text{NB}(n) \cap U}|\mathbf{x}_V)$ yields comparable results with multiple samples. Therefore, in the experiments, only one sample is used for efficiency purpose.

Given Eq. (8) and (9), $q^*(\mathbf{y}_n|\mathbf{x}_V)$ can thus be approximated as $q^*(\mathbf{y}_n|\mathbf{x}_V) \approx p_\phi(\mathbf{y}_n|\hat{\mathbf{y}}_{\text{NB}(n)}, \mathbf{x}_V)$. Therefore, we could instead treat $p_\phi(\mathbf{y}_n|\hat{\mathbf{y}}_{\text{NB}(n)}, \mathbf{x}_V)$ as target, and minimize $\text{KL}(p_\phi(\mathbf{y}_n|\hat{\mathbf{y}}_{\text{NB}(n)}, \mathbf{x}_V)||q_\theta(\mathbf{y}_n|\mathbf{x}_V))$. We further use a parallel update strategy (Koller & Friedman, 2009) to speed up training, where we jointly optimize $q_\theta(\mathbf{y}_n|\mathbf{x}_V)$ for every unlabeled object n , yielding the objective as follows:

$$O_{\theta,U} = \sum_{n \in U} \mathbb{E}_{p_\phi(\mathbf{y}_n|\hat{\mathbf{y}}_{\text{NB}(n)}, \mathbf{x}_V)} [\log q_\theta(\mathbf{y}_n|\mathbf{x}_V)]. \quad (10)$$

Besides, we notice that q_θ can be also trained by predicting the labels for the labeled objects. Therefore, we also let q_θ maximize the following supervised objective function:

$$O_{\theta,L} = \sum_{n \in L} \log q_\theta(\mathbf{y}_n|\mathbf{x}_V). \quad (11)$$

Here, \mathbf{y}_n is the ground-truth label of n . By adding Eq. (10) and (11), we obtain the overall objective for optimizing θ :

$$O_\theta = O_{\theta,U} + O_{\theta,L}. \quad (12)$$

4.3. Learning

In the M-step, we seek to learn the parameter ϕ . More specifically, we will fix q_θ and further update p_ϕ to maximize Eq. (5). With the objective function, we notice that only the conditional distribution $p_\phi(\mathbf{y}_n|\mathbf{y}_{\text{NB}(n)}, \mathbf{x}_V)$ is required for p_ϕ in both the inference and learning steps (Eq. (10) and (5)). Therefore, instead of defining the joint distribution of object labels $p_\phi(\mathbf{y}_V|\mathbf{x}_V)$ by specifying the potential function, we can simply focus on modeling the conditional distribution. Here, we parameterize the conditional distribution $p_\phi(\mathbf{y}_n|\mathbf{y}_{\text{NB}(n)}, \mathbf{x}_V)$ with another non-linear graph neural network model (GNN) because of its effectiveness:

$$p_\phi(\mathbf{y}_n|\mathbf{y}_{\text{NB}(n)}, \mathbf{x}_V) = \text{Cat}(\mathbf{y}_n|\text{softmax}(W_\phi \mathbf{h}_{\phi,n})). \quad (13)$$

Here, the distribution of \mathbf{y}_n is characterized by a softmax classifier, which takes the object representation $\mathbf{h}_{\phi,n}$ learned by a GNN model as features, and we denote the GNN as GNN_ϕ . When learning the object representation $\mathbf{h}_{\phi,n}$, GNN_ϕ treats all the labels $\mathbf{y}_{\text{NB}(n)}$ surrounding the object n as features. Therefore, GNN_ϕ essentially models local dependencies of object labels. With the above formulation, we no longer require any hand-crafted feature functions.

The framework is related to the label propagation methods (Zhu et al., 2003; Zhou et al., 2004), which also update each object label by combining the surrounding labels. However, these methods propagate labels in a fixed and linear way, whereas GNN_ϕ is in a learnable and non-linear way.

One notable thing is that when defining $p_\phi(\mathbf{y}_n|\mathbf{y}_{\text{NB}(n)}, \mathbf{x}_V)$, GNN_ϕ only uses the object labels $\mathbf{y}_{\text{NB}(n)}$ surrounding the object n as features, but GNN_ϕ is flexible to incorporate other features. For example, we can follow existing SRL

Algorithm 1 Optimization Algorithm

Input: A graph G , some labeled objects (L, \mathbf{y}_L) .

Output: Object labels \mathbf{y}_U for unlabeled objects U .

Pre-train q_θ with \mathbf{y}_L according to Eq. (11).

while not converge **do**

 □ *M-Step: Learning Procedure*

 Annotate unlabeled objects with q_θ .

 Denote the sampled labels as $\hat{\mathbf{y}}_U$.

 Set $\hat{\mathbf{y}}_V = (\mathbf{y}_L, \hat{\mathbf{y}}_U)$ and update p_ϕ with Eq. (14).

 □ *E-Step: Inference Procedure*

 Annotate unlabeled objects with p_ϕ and $\hat{\mathbf{y}}_V$.

 Denote the predicted label distribution as $p_\phi(\mathbf{y}_U)$.

 Update q_θ with Eq. (10), (11) based on $p_\phi(\mathbf{y}_U), \mathbf{y}_L$.

end while

Classify each unlabeled object n based on $q_\theta(\mathbf{y}_n|\mathbf{x}_V)$.

methods, and take both the surrounding object labels $\mathbf{y}_{\text{NB}(n)}$ and surrounding attributes $\mathbf{x}_{\text{NB}(n)}$ as features in GNN_ϕ . We will discuss this variant in our experiment (see Sec. 6.4).

Another thing is that based on the overall formulation of p_ϕ , i.e. Eq. (1), each object label \mathbf{y}_n should only depend on its adjacent object labels $\mathbf{y}_{\text{NB}(n)}$ and object attributes \mathbf{x}_V , which implies GNN_ϕ should not have more than one message passing layer. However, a common practice in the literature of graph neural networks is to use multiple message passing layers during training, which can well model the long-range dependency between different objects. Therefore, we also explore using multiple message passing layers to capture such long-range dependency (see Sec. 6.4).

When optimizing p_ϕ to maximize Eq. (5), we estimate the expectation in Eq. (5) by drawing a sample from $q_\theta(\mathbf{y}_U|\mathbf{x}_V)$. More specifically, if n is an unlabeled object, then we sample $\hat{\mathbf{y}}_n \sim q_\theta(\mathbf{y}_n|\mathbf{x}_V)$, and otherwise we set $\hat{\mathbf{y}}_n$ as the ground-truth label. Afterwards, the parameter ϕ can be optimized by maximizing the following objective function:

$$O_\phi = \sum_{n \in V} \log p_\phi(\hat{\mathbf{y}}_n|\hat{\mathbf{y}}_{\text{NB}(n)}, \mathbf{x}_V). \quad (14)$$

4.4. Optimization

To optimize our approach, we pre-train q_θ with the labeled objects. Then we alternatively optimize p_ϕ and q_θ until convergence. Afterwards, both p_ϕ and q_θ can be used to classify unlabeled objects. In practice, q_θ consistently outperforms p_ϕ , and thus we use q_θ to infer object labels by default. We summarize the detailed optimization algorithm in Alg. 1.

Fig. 1 presents an illustration of the framework. For the central object, q_θ uses the attributes of its surrounding objects to learn its representation, and further predicts the label. In contrast, p_ϕ utilizes the labels of the surrounding objects as features. If a neighbor is unlabeled, we simply use a label sampled from q_θ instead. In the E-step, p_ϕ predicts the

label for the central object, which is then treated as target to update q_θ . In the M-step, q_θ predicts the label for the central object, which serves as the target data to update p_ϕ .

5. Application

Besides semi-supervised object classification, our approach can also be naturally extended to many other tasks. In this section, we choose two applications for demonstration.

5.1. Unsupervised Node Representation Learning

Though GMNN is designed to learn node representations for semi-supervised node classification, it can be naturally extended to learn node representation without any labeled nodes. In this case, as there are no labeled nodes, we instead predict the neighbors for each node. In this way, the neighbors of each node are treated as the pseudo labels, and this idea is widely used in existing studies (Perozzi et al., 2014; Tang et al., 2015; Grover & Leskovec, 2016). Based on that, we first train the inference network q_θ with the pseudo label of each node, then we continue training with Alg. 1, in which all the nodes labels are treated as unobserved variables. In the E-step, we infer the neighbor distribution for each node with q_θ by optimizing Eq. (10), during which p_ϕ encourages the inferred neighbor distributions to be locally smooth. In the M-step, we update p_ϕ with Eq. (14) to model the local dependency of the inferred neighbor distributions.

The inference network q_θ in the above method shares similar ideas with existing methods for unsupervised node representation learning (Perozzi et al., 2014; Tang et al., 2015; Grover & Leskovec, 2016), since they all seek to predict the neighbors for each node. However, we also use p_ϕ to ensure that the neighbor distributions inferred by q_θ are locally smooth, and thus p_ϕ serves as a regularizer to improve q_θ .

5.2. Link Classification

GMNN can be also applied to link classification. Given a few labeled links, the goal is to classify the remaining links.

Following the idea in existing SRL studies (Taskar et al., 2004), we reformulate link classification as an object classification problem. Specifically, we construct a line graph \tilde{G} from the original object graph G . The object set \tilde{V} in the line graph corresponds to the link set E in the original graph. Two objects are linked in the line graph if their corresponding links in the original graph share a node. The attributes of each node in the line graph is defined as the nodes of the corresponding link in the original graph.

Based on that, the link classification task on the original graph is formulated as the object classification task on the line graph. Therefore, we can apply our object classification approach to the line graph for solving the original problem.

6. Experiment

In this section, we evaluate the performance of GMNN on three tasks, including object classification, unsupervised node representation learning, and link classification.

6.1. Datasets and Experiment Settings

For object classification, we follow existing studies (Yang et al., 2016; Kipf & Welling, 2017; Veličković et al., 2018) and use three benchmark datasets from Sen et al. (2008) for evaluation, including Cora, Citeseer, Pubmed. In each dataset, 20 objects from each class are treated as labeled objects, and we use the same data partition as in Yang et al. (2016). Accuracy is used as the evaluation metric.

For unsupervised node representation learning, we also use the above three datasets, in which objects are treated as nodes. We learn node representations without using any labeled nodes. To evaluate the learned representations, we follow Veličković et al. (2019) and treat the representations as features to train a linear classifier on the labeled nodes. Then we classify the test nodes and report the accuracy. Note that we use the same data partition as in object classification.

For link classification, we construct two datasets from the Bitcoin Alpha and the Bitcoin OTC datasets (Kumar et al., 2016; 2018) respectively. The datasets contain graphs between Bitcoin users, and the weight of a link represents the trust degree of connected users. We treat links with weights greater than 3 as positive instances, and links with weights less than -3 are treated as negative ones. Given a few labeled links, We try to classify the test links. As the positive and negative links are quite unbalanced, we report the F1 score.

6.2. Compared Algorithms

GNN Methods. For object classification and link classification, we mainly compare with the recently-proposed Graph Convolutional Network (Kipf & Welling, 2017) and Graph Attention Network (Veličković et al., 2018). However, GAT cannot scale up to both datasets in the link classification task, so the performance is not reported. For unsupervised node representation learning, we compare with Deep Graph Infomax (Veličković et al., 2019), which is the state-of-the-art method. Besides, we also compare with DeepWalk (Perozzi et al., 2014) and Planetoid (Yang et al., 2016).

SRL Methods. For SRL methods, we compare with the Probabilistic Relational Model (Koller & Pfeffer, 1998), the Relational Markov Network (Taskar et al., 2002) and the Markov Logic Network (Richardson & Domingos, 2006). In the PRM method, we assume that the label of an object depends on the attributes of itself, its adjacent objects, and the labels of its adjacent objects. For the RMN and MLN methods, we follow Taskar et al. (2002) and use a logistic

Table 1. Dataset statistics. OC, NRL, LC represent object classification, node representation learning and link classification respectively.

| Dataset | Task | # Nodes | # Edges | # Features | # Classes | # Training | # Validation | # Test |
|---------------|----------|---------|---------|------------|-----------|------------|--------------|--------|
| Cora | OC / NRL | 2,708 | 5,429 | 1,433 | 7 | 140 | 500 | 1,000 |
| Citeseer | OC / NRL | 3,327 | 4,732 | 3,703 | 6 | 120 | 500 | 1,000 |
| Pubmed | OC / NRL | 19,717 | 44,338 | 500 | 3 | 60 | 500 | 1,000 |
| Bitcoin Alpha | LC | 3,783 | 24,186 | 3,783 | 2 | 100 | 500 | 3,221 |
| Bitcoin OTC | LC | 5,881 | 35,592 | 5,881 | 2 | 100 | 500 | 5,947 |

Table 2. Results of object classification (%). [*] means the results are taken from the corresponding papers.

| Category | Algorithm | Cora | Citeseer | Pubmed |
|----------|------------------------|-------------|-------------|-------------|
| SSL | LP | 74.2 | 56.3 | 71.6 |
| | PRM | 77.0 | 63.4 | 68.3 |
| SRL | RMN | 71.3 | 68.0 | 70.7 |
| | MLN | 74.6 | 68.0 | 75.3 |
| GNN | Planetoid * | 75.7 | 64.7 | 77.2 |
| | GCN * | 81.5 | 70.3 | 79.0 |
| | GAT * | 83.0 | 72.5 | 79.0 |
| GMNN | W/o Attr. in p_ϕ | 83.4 | 73.1 | 81.4 |
| | With Attr. in p_ϕ | 83.7 | 72.9 | 81.8 |
| | Best results | 83.7 | 73.6 | 81.9 |

Table 3. Results of unsupervised node representation learning (%). [*] means the results are taken from corresponding papers.

| Category | Algorithm | Cora | Citeseer | Pubmed |
|----------|------------------------------|-------------|-------------|-------------|
| GNN | DeepWalk * | 67.2 | 43.2 | 65.3 |
| | DGI * | 82.3 | 71.8 | 76.8 |
| GMNN | With only q_θ | 78.1 | 68.0 | 79.3 |
| | With q_θ and p_ϕ | 82.8 | 71.5 | 81.6 |

regression model locally for each object. This logistic regression model takes the attributes of each object and also those of its neighbors as features. Besides, we treat the labels of two linked objects as a clique template, which is the same as in Taskar et al. (2002). In RMN, a complete score table is employed for modeling label dependency, which maintains a potential score for every possible combination of object labels in a clique. In MLN, we simply use one indicator function in the potential function, and the indicator function judges whether the objects in a clique have the same label. Loop belief propagation (Murphy et al., 1999) is used for approximation inference in RMN and MLN.

SSL Methods. For methods under the category of graph-based semi-supervised classification, we choose the label propagation method (Zhou et al., 2004) to compare with.

6.3. Parameter Settings

Object Classification. For GMNN, p_ϕ and q_θ are composed of two graph convolutional layers with 16 hidden units and ReLU activation (Nair & Hinton, 2010), followed by the softmax function, as in Kipf & Welling (2017). We reweight each feature of a node to 1 if the original weight is greater than 0. Dropout (Srivastava et al., 2014) is applied to the network inputs with $p = 0.5$. We use the RMSProp optimizer (Tieleman & Hinton, 2012) during training, with the initial learning rate as 0.05 and weight decay as 0.0005. In each iteration, both networks are trained for 100 epochs.

The mean accuracy over 100 runs is reported in experiment.

Unsupervised Node Representation Learning. For the GMNN approach, p_ϕ and q_θ are composed of two graph convolutional layers followed by a linear layer and the softmax function. The dimension of hidden layers is set as 512 for Cora and Citeseer, and 256 for Pubmed, which are the same as in Veličković et al. (2019). ReLU (Nair & Hinton, 2010) is used as the activation function. Each node feature is reweighted to 1 if the original weight is larger than 0. We apply dropout (Srivastava et al., 2014) to the inputs of both networks with $p = 0.5$. The Adam SGD optimizer (Kingma & Ba, 2014) is used for training, with initial learning rate as 0.1 and weight decay as 0.0005. We empirically train q_θ for 200 epochs during pre-training. Afterwards, we train both p_ϕ and q_θ for 2 iterations, with 100 epochs for each network per iteration. For the Pubmed dataset, we transform the raw node features into binary value since it can result in better performance. The mean accuracy over 50 runs is reported.

Link Classification. The setting of GMNN in this task is similar as in object classification, with the following differences. The dimension of the hidden layers is set as 128. No weight decay and dropout are used. In each iteration, both networks are trained for 5 epochs with the Adam optimizer (Kingma & Ba, 2014), and the learning rate is 0.01.

6.4. Results

1. Comparison with the Baseline Methods. The quantitative results on the three tasks are presented in Tab. 2, 3, 4 respectively. For object classification, GMNN significantly outperforms all the SRL methods. The performance gain is from two folds. First, during inference, GMNN employs a GNN model, which can learn effective object representations to improve inference. Second, during learning, we model the local label dependency with another GNN, which is more effective compared with SRL methods. GMNN is also superior to the label propagation method, as GMNN is able to use object attributes and propagate labels in a non-linear way. Compared with GCN, which employs the same architecture as the inference network in GMNN, GMNN significantly outperforms GCN, and the performance gain mainly comes from the capability of modeling label dependencies. Besides, GMNN also outperforms GAT, but their performances are quite close. This is because GAT utilizes a much more complicated architecture. Since GAT is less efficient, it is not used in GMNN, but we anticipate the re-

sults can be further improved by using GAT, and we leave it as future work. In addition, by incorporating the object attributes in the learning network p_ϕ , we further improve the performance, showing that GMNN is flexible and also effective to incorporate additional features in the learning network. For link classification, we obtain similar results.

For unsupervised node representation learning, GMNN achieves state-of-the-art results on the Cora and Pubmed datasets. The reason is that GMNN effectively models the smoothness of the neighbor distributions for different nodes with the p_ϕ network. Besides, the performance of GMNN is quite close to the performance in the semi-supervised setting (Tab. 2), showing that the learned representations are quite effective. We also compare with a variant without using the p_ϕ network (with only q_θ). In this case, we see that the performance drops significantly, showing the importance of using p_ϕ as a regularizer over the neighbor distributions.

Table 4. Results of link classification (%).

| Category | Algorithm | Bitcoin Alpha | Bitcoin OTC |
|----------|------------------------|---------------|--------------|
| SSL | LP | 59.68 | 65.58 |
| | PRM | 58.59 | 64.37 |
| SRL | RMN | 59.56 | 65.59 |
| | MLN | 60.87 | 65.62 |
| GNN | DeepWalk | 62.71 | 63.20 |
| | GCN | 64.00 | 65.69 |
| GMNN | W/o Attr. in p_ϕ | 65.59 | 66.62 |
| | With Attr. in p_ϕ | 65.86 | 66.83 |

2. Analysis of the Amortized Inference. In GMNN, we employ amortized inference, and parameterize the posterior label distribution by using a GNN model. In this section, we thoroughly look into this strategy, and present some analysis in Tab. 5. Here, the variant “Non-amortized” simply models each $q_\theta(\mathbf{y}_n|\mathbf{x}_V)$ as a categorical distribution with independent parameters, and performs fix-point iteration (i.e. Eq. (8)) to calculate the value. We see that the performance of this variant is very poor on all datasets. By parameterizing the posterior distribution as a neural network, which leverages the own attributes of each object for inference, the performance (see “1 Linear Layer”) is significantly improved, but still not satisfactory. With several GC layers, we are able to incorporate the attributes from the surrounding neighbors for each object, yielding further significant improvement. The above observations prove the effectiveness of our strategy for inferring the posterior label distributions.

Table 5. Analysis of amortized inference (%).

| Architecture | Cora | Citeseer | Pubmed |
|----------------|------|----------|--------|
| Non-amortized | 45.3 | 28.1 | 42.2 |
| 1 Linear Layer | 55.8 | 57.5 | 69.8 |
| 1 GC Layer | 72.9 | 67.6 | 71.8 |
| 2 GC Layers | 83.4 | 73.1 | 81.4 |
| 3 GC Layers | 82.0 | 70.6 | 80.7 |

3. Ablation Study of the Learning Network. In GMNN, the conditional distribution $p_\phi(\mathbf{y}_n|\mathbf{y}_{\text{NB}(n)}, \mathbf{x}_V)$ is parameterized as another GNN, which essentially models the local label dependency. In this section, we compare different architectures of the GNN on the object classification task, and the results are presented in Tab. 6. Here, the variant “1 Mean Pooling Layer” computes the distribution of \mathbf{y}_n as the linear combination of $\{\mathbf{y}_{n'}\}_{n' \in \text{NB}(n)}$. This variant is similar to label propagation methods, and its performance is quite competitive. However, the weights of different neighbors during propagation are fixed. By parameterizing the conditional distribution with several GC layers, we are able to automatically learn the propagation weights, and thus obtain superior results on all datasets. This observation proves the effectiveness of employing GNNs in the learning procedure.

Table 6. Ablation study of the learning network (%).

| Architecture | Cora | Citeseer | Pubmed |
|----------------------|------|----------|--------|
| 1 Mean Pooling Layer | 82.4 | 71.9 | 80.7 |
| 1 GC Layer | 83.1 | 73.1 | 80.9 |
| 2 GC Layers | 83.4 | 73.1 | 81.4 |
| 3 GC Layers | 83.6 | 73.0 | 81.5 |

4. Convergence Analysis. In GMNN, we utilize the variational EM algorithm for optimization, which consists of an E-step and an M-step in each iteration. Next, we analyze the convergence of GMNN. We take the Cora and Citeseer datasets on object classification as examples, and report the validation accuracy of both the q_θ and p_ϕ networks at each iteration. Fig. 2 presents the convergence curve, in which iteration 0 corresponds to the pre-training stage. GMNN takes only few iterations to convergence, which is very efficient.

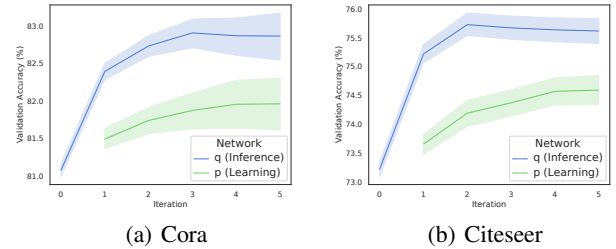


Figure 2. Convergence analysis.

7. Conclusion

This paper studies semi-supervised object classification, which is a fundamental problem in relational data modeling, and a novel approach called the GMNN is proposed. GMNN employs a conditional random field to model the joint distribution of object labels, and two graph neural networks are utilized to improve both the inference and learning procedures. Experimental results on three tasks prove the effectiveness of GMNN. In the future, we plan to further improve GMNN to deal with graphs with multiple edge types, such as knowledge graphs (Bollacker et al., 2008).

Acknowledgements

We would like to thank all the anonymous reviewers for their insightful comments. We also thank Carlos Lassance, Mingzhe Wang, Zhaocheng Zhu, Weiping Song for reviewing the paper before submission. Jian Tang is supported by the Natural Sciences and Engineering Research Council of Canada, as well as the Canada CIFAR AI Chair Program.

References

- Besag, J. Statistical analysis of non-lattice data. *The statistician*, pp. 179–195, 1975.
- Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, 2008.
- Dai, H., Dai, B., and Song, L. Discriminative embeddings of latent variable models for structured data. In *ICML*, 2016.
- Deng, Y., Kim, Y., Chiu, J., Guo, D., and Rush, A. Latent alignment and variational attention. In *NeurIPS*, 2018.
- Dettmers, T., Minervini, P., Stenetorp, P., and Riedel, S. Convolutional 2d knowledge graph embeddings. In *AAAI*, 2018.
- Friedman, N., Getoor, L., Koller, D., and Pfeffer, A. Learning probabilistic relational models. In *IJCAI*, 1999.
- Gershman, S. and Goodman, N. Amortized inference in probabilistic reasoning. In *CogSci*, 2014.
- Getoor, L., Friedman, N., Koller, D., and Taskar, B. Learning probabilistic models of relational structure. In *ICML*, 2001a.
- Getoor, L., Segal, E., Taskar, B., and Koller, D. Probabilistic models of text and link structure for hypertext classification. In *IJCAI Workshop*, 2001b.
- Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *ICML*, 2017.
- Grover, A. and Leskovec, J. node2vec: Scalable feature learning for networks. In *KDD*, 2016.
- Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *NIPS*, 2017.
- Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR*, 2014.
- Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *ICLR*, 2014.
- Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.
- Kok, S. and Domingos, P. Learning the structure of markov logic networks. In *ICML*, 2005.
- Koller, D. and Friedman, N. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.
- Koller, D. and Pfeffer, A. Probabilistic frame-based systems. In *AAAI/IAAI*, 1998.
- Kumar, S., Spezzano, F., Subrahmanian, V., and Faloutsos, C. Edge weight prediction in weighted signed networks. In *ICDM*, 2016.
- Kumar, S., Hooi, B., Makhija, D., Kumar, M., Faloutsos, C., and Subrahmanian, V. Rev2: Fraudulent user prediction in rating platforms. In *WSDM*, 2018.
- Lafferty, J., McCallum, A., and Pereira, F. C. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.
- Murphy, K. P., Weiss, Y., and Jordan, M. I. Loopy belief propagation for approximate inference: An empirical study. In *UAI*, 1999.
- Nair, V. and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.
- Neal, R. M. and Hinton, G. E. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pp. 355–368. Springer, 1998.
- Opper, M. and Saad, D. *Advanced mean field methods: Theory and practice*. MIT press, 2001.
- Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk: Online learning of social representations. In *KDD*, 2014.
- Richardson, M. and Domingos, P. Markov logic networks. *Machine learning*, 62(1-2):107–136, 2006.
- Salakhutdinov, R. and Larochelle, H. Efficient learning of deep boltzmann machines. In *AISTATS*, 2010.
- Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. Collective classification in network data. *AI magazine*, 29(3):93, 2008.
- Singla, P. and Domingos, P. Discriminative training of markov logic networks. In *AAAI*, 2005.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

- Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. Line: Large-scale information network embedding. In *WWW*, 2015.
- Taskar, B., Abbeel, P., and Koller, D. Discriminative probabilistic models for relational data. In *UAI*, 2002.
- Taskar, B., Wong, M.-F., Abbeel, P., and Koller, D. Link prediction in relational data. In *NIPS*, 2004.
- Taskar, B., Abbeel, P., Wong, M.-F., and Koller, D. Relational markov networks. *Introduction to statistical relational learning*, 2007.
- Tieleman, T. and Hinton, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4 (2):26–31, 2012.
- Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *ICLR*, 2018.
- Veličković, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D. Deep graph infomax. In *ICLR*, 2019.
- Xiang, R. and Neville, J. Pseudolikelihood em for within-network relational learning. In *ICDM*, 2008.
- Yang, Z., Cohen, W., and Salakhudinov, R. Revisiting semi-supervised learning with graph embeddings. In *ICML*, 2016.
- Yoon, K., Liao, R., Xiong, Y., Zhang, L., Fetaya, E., Urtasun, R., Zemel, R., and Pitkow, X. Inference in probabilistic graphical models by graph neural networks. In *ICLR Workshop*, 2018.
- Zhou, D., Bousquet, O., Lal, T. N., Weston, J., and Schölkopf, B. Learning with local and global consistency. In *NIPS*, 2004.
- Zhu, X., Ghahramani, Z., and Lafferty, J. D. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, 2003.

A. Optimality Condition for q_θ

Theorem A.1. Given an object n , consider a fixed variational distribution $q_\theta(\mathbf{y}_{NB(n) \cap U} | \mathbf{x}_V)$ for nodes in $NB(n) \cap U$. Then the optimum of $q_\theta(\mathbf{y}_n | \mathbf{x}_V)$, denoted by $q^*(\mathbf{y}_n | \mathbf{x}_V)$, is characterized by the following condition:

$$\log q^*(\mathbf{y}_n | \mathbf{x}_V) = \mathbb{E}_{q_\theta(\mathbf{y}_{NB(n) \cap U} | \mathbf{x}_V)} [\log p_\phi(\mathbf{y}_n | \mathbf{y}_{NB(n)}, \mathbf{x}_V)] + \text{const}.$$

Proof. To make the notation more concise, we will omit \mathbf{x}_V in the following proof (e.g. simplifying $q_\theta(\mathbf{y}_n | \mathbf{x}_V)$ as $q_\theta(\mathbf{y}_n)$). Recall that our overall goal for $q_\theta(\mathbf{y}_n)$ is to minimize the KL divergence between $q_\theta(\mathbf{y}_U)$ and $p_\phi(\mathbf{y}_U | \mathbf{y}_L)$. Therefore, the objective function for $q_\theta(\mathbf{y}_n)$ could be formulated as follows:

$$\begin{aligned} O(q_\theta(\mathbf{y}_n)) &= -\text{KL}(q_\theta(\mathbf{y}_U) || p_\phi(\mathbf{y}_U | \mathbf{y}_L)) \\ &= \sum_{\mathbf{y}_U} q_\theta(\mathbf{y}_U) [\log p_\phi(\mathbf{y}_U | \mathbf{y}_L) - \log q_\theta(\mathbf{y}_U)] \\ &= \sum_{\mathbf{y}_U} \left(\prod_{n'} q_\theta(\mathbf{y}_{n'}) \right) \left[\log p_\phi(\mathbf{y}_U, \mathbf{y}_L) - \sum_{n'} \log q_\theta(\mathbf{y}_{n'}) \right] + \text{const} \\ &= \sum_{\mathbf{y}_n} \sum_{\mathbf{y}_{U \setminus n}} \left(q_\theta(\mathbf{y}_n) \prod_{n' \neq n} q_\theta(\mathbf{y}_{n'}) \right) \left[\log p_\phi(\mathbf{y}_U, \mathbf{y}_L) - \sum_{n'} \log q_\theta(\mathbf{y}_{n'}) \right] + \text{const} \\ &= \sum_{\mathbf{y}_n} q_\theta(\mathbf{y}_n) \sum_{\mathbf{y}_{U \setminus n}} \prod_{n' \neq n} q_\theta(\mathbf{y}_{n'}) \log p_\phi(\mathbf{y}_U, \mathbf{y}_L) - \\ &\quad \sum_{\mathbf{y}_n} q_\theta(\mathbf{y}_n) \sum_{\mathbf{y}_{U \setminus n}} \prod_{n' \neq n} q_\theta(\mathbf{y}_{n'}) \left[\sum_{n' \neq n} \log q_\theta(\mathbf{y}_{n'}) + \log q_\theta(\mathbf{y}_n) \right] + \text{const} \\ &= \sum_{\mathbf{y}_n} q_\theta(\mathbf{y}_n) \log \mathcal{F}(\mathbf{y}_n) - \sum_{\mathbf{y}_n} q_\theta(\mathbf{y}_n) \log q_\theta(\mathbf{y}_n) + \text{const} \\ &= -\text{KL} \left(q_\theta(\mathbf{y}_n) || \frac{\mathcal{F}(\mathbf{y}_n)}{Z} \right) + \text{const}. \end{aligned}$$

Here, Z is a normalization term, which makes $\mathcal{F}(\mathbf{y}_n)$ a valid distribution on \mathbf{y}_n , and we have:

$$\log \mathcal{F}(\mathbf{y}_n) = \sum_{\mathbf{y}_{U \setminus n}} \prod_{n' \neq n} q_\theta(\mathbf{y}_{n'}) \log p_\phi(\mathbf{y}_U, \mathbf{y}_L) = \mathbb{E}_{q_\theta(\mathbf{y}_{U \setminus n})} [\log p_\phi(\mathbf{y}_U, \mathbf{y}_L)].$$

Based on the above mathematical manipulation of the objective function $O(q_\theta(\mathbf{y}_n))$, if a local optimal of $q_\theta(\mathbf{y}_n)$ is denoted by $q^*(\mathbf{y}_n | \mathbf{x}_V)$, then $q^*(\mathbf{y}_n | \mathbf{x}_V)$ must be equal to $\frac{\mathcal{F}(\mathbf{y}_n)}{Z}$, and thus we have:

$$\begin{aligned} \log q^*(\mathbf{y}_n) &= \log \mathcal{F}(\mathbf{y}_n) + \text{const} \\ &= \mathbb{E}_{q_\theta(\mathbf{y}_{U \setminus n})} [\log p_\phi(\mathbf{y}_U, \mathbf{y}_L)] + \text{const} \\ &= \mathbb{E}_{q_\theta(\mathbf{y}_{U \setminus n})} [\log p_\phi(\mathbf{y}_n | \mathbf{y}_{V \setminus n})] + \text{const} \\ &= \mathbb{E}_{q_\theta(\mathbf{y}_{U \setminus n})} [\log p_\phi(\mathbf{y}_n | \mathbf{y}_{NB(n)})] + \text{const} \\ &= \mathbb{E}_{q_\theta(\mathbf{y}_{NB(n) \cap U})} [\log p_\phi(\mathbf{y}_n | \mathbf{y}_{NB(n)})] + \text{const}. \end{aligned}$$

Here, $p_\phi(\mathbf{y}_n | \mathbf{y}_{V \setminus n}) = p_\phi(\mathbf{y}_n | \mathbf{y}_{NB(n)})$ is based on the conditional independence property of Markov networks. \square

B. Additional Experiment

B.1. Results on Random Data Splits

In the previous experiment, we have seen that GMNN significantly outperforms all the baseline methods for semi-supervised object classification under the data splits from [Yang et al. \(2016\)](#). To further validate the effectiveness of GMNN, we also

Table 7. Results on random data splits (%).

| Algorithm | Cora | Citeseer | Pubmed |
|-----------|-------------|-------------|-------------|
| GCN | 81.5 | 71.3 | 80.3 |
| GAT | 82.1 | 71.5 | 80.1 |
| GMNN | 83.1 | 73.0 | 81.9 |

evaluate GMNN on some random data splits. Specifically, we randomly create 10 data splits for each dataset. The size of the training, validation and test sets in each split is the same as the split in Yang et al. (2016). We compare GMNN with GCN (Kipf & Welling, 2017) and GAT (Veličković et al., 2018) on those random data splits, as they are the most competitive baseline methods. For each data split, we run each method with 10 different seeds, and report the overall mean accuracy in Tab. 7. We see GMNN consistently outperforms GCN and GAT on all datasets, proving the effectiveness of GMNN.

B.2. Results on Few-shot Learning Settings

Table 8. Results on few-shot learning settings (%).

| Algorithm | Cora | Citeseer | Pubmed |
|-----------|-------------|-------------|-------------|
| GCN | 74.9 | 69.0 | 76.9 |
| GAT | 77.0 | 68.9 | 75.4 |
| GMNN | 78.6 | 72.7 | 79.1 |

In the previous experiment, we have proved the effectiveness of GMNN for object classification in the semi-supervised setting. Next, we further conduct experiment in the few-shot learning setting to evaluate the robustness of GMNN to data sparsity. We choose GCN and GAT for comparison. For each dataset, we randomly sample 5 labeled nodes under each class as training data, and run each method with 100 different seeds. The mean accuracy is shown in Tab. 8. We see GMNN significantly outperforms GCN and GAT. The improvement is even larger than the case of semi-supervised setting, where 20 labeled nodes under each class are used for training. This result proves that GMNN is robust to the sparsity of training data.

B.3. Comparison with Self-training Methods

Table 9. Comparison with self-training methods (%).

| Algorithm | Cora | Citeseer | Pubmed |
|---------------|-------------|-------------|-------------|
| Self-training | 82.7 | 72.4 | 80.1 |
| GMNN | 83.4 | 73.1 | 81.4 |

Our proposed GMNN approach is related to self-training frameworks. In GMNN, the p_ϕ network essentially tries to annotate unlabeled objects, and the annotated objects are further treated as extra data to update q_θ through Eq. (10). Similarly, in self-training, we typically use q_θ itself to annotate unlabeled objects, and collect extra training data for q_θ . Next, we compare GMNN with the self-training method for semi-supervised object classification, and the results are presented in Tab. 9.

We see GMNN consistently outperforms the self-training method. The reason is that the self-training method uses q_θ for both inference and annotation, while GMNN uses two different networks q_θ and p_ϕ to collaborate with each other. The information captured by q_θ and p_ϕ is complementary, and therefore GMNN achieves much better results.

B.4. Comparison of Different Approximation Methods

In GMNN, we use a mean-field variational distribution q_θ for inference, and the optimal q_θ is given by the fixed-point condition in Eq. (8). Learning the optimal q_θ requires computing the right-hand side of Eq. (8), which involves the expectation with respect to $q_\theta(\mathbf{y}_{\text{NB}(n) \cap U} | \mathbf{x}_V)$ for each node n . To estimate the expectation, we notice that $q_\theta(\mathbf{y}_{\text{NB}(n) \cap U} | \mathbf{x}_V)$ can be factorized as $\prod_{n' \in \text{NB}(n) \cap U} q_\theta(\mathbf{y}_{n'} | \mathbf{x}_V)$. Based on that, we can develop several empirical approximation methods.

Single Sample. The simplest way is to draw a single sample $\hat{\mathbf{y}}_{n'} \sim q_\theta(\mathbf{y}_{n'} | \mathbf{x}_V)$ for each node $n' \in \text{NB}(n) \cap U$, and then we could use the sample to estimate the expectation.

Table 10. Comparison of different approximation methods (%).

| Method | Cora | Citeseer | Pubmed |
|------------------|-------------|-------------|-------------|
| Single Sample | 82.1 | 71.5 | 80.4 |
| Multiple Samples | 83.2 | 72.5 | 81.1 |
| Annealing | 83.4 | 73.1 | 81.4 |
| Max Pooling | 83.2 | 72.8 | 81.2 |
| Mean Pooling | 83.4 | 72.6 | 80.5 |

Multiple Samples. In practice, we can also draw multiple samples from $q_\theta(\mathbf{y}_{n'}|\mathbf{x}_V)$ for each node $n' \in \text{NB}(n) \cap U$ to estimate the expectation. Such a method has lower variance but entails higher cost.

Annealing. Another method is to introduce an annealing parameter τ in $q_\theta(\mathbf{y}_{n'}|\mathbf{x}_V)$, so that we have:

$$q_\theta(\mathbf{y}_{n'}|\mathbf{x}_V) = \text{Cat}(\mathbf{y}_{n'}|\text{softmax}(\frac{W_\theta \mathbf{h}_{\theta, n'}}{\tau})).$$

Then we can set τ to a small value (e.g. 0.1) and draw a sample $\hat{\mathbf{y}}_{n'} \sim q_\theta(\mathbf{y}_{n'}|\mathbf{x}_V)$ for $n' \in \text{NB}(n) \cap U$ to estimate the expectation, which typically has lower variance.

Max Pooling. Another method is max pooling, where we set $\hat{\mathbf{y}}_{n'} = \arg \max_{\mathbf{y}_{n'}} q_\theta(\mathbf{y}_{n'}|\mathbf{x}_V)$ for $n' \in \text{NB}(n) \cap U$, and use $\{\hat{\mathbf{y}}_{n'}\}_{n' \in \text{NB}(n) \cap U}$ as a sample to estimate the expectation.

Mean Pooling. Besides, we can also use the mean pooling method similar to the soft attention method in Deng et al. (2018). Specifically, suppose that we have C classes for classification, then the label of each node can be viewed as a one-hot C -dimensional vector. Based on that, we set $\bar{\mathbf{y}}_{n'} = \mathbb{E}_{q_\theta(\mathbf{y}_{n'}|\mathbf{x}_V)}[\mathbf{y}_{n'}]$ for each unlabeled neighbor n' of the object n , which can be understood as a soft label vector of that neighbor. For each labeled neighbor n' of the object n , we set $\bar{\mathbf{y}}_{n'}$ as the one-hot label vector. Then we can approximate the expectation as:

$$\begin{aligned} & \mathbb{E}_{q_\theta(\mathbf{y}_{\text{NB}(n) \cap U}|\mathbf{x}_V)}[\log p_\phi(\mathbf{y}_n|\mathbf{y}_{\text{NB}(n)}, \mathbf{x}_V)] \\ & \approx p_\phi(\mathbf{y}_n|\bar{\mathbf{y}}_{\text{NB}(n)}, \mathbf{x}_V) \stackrel{\text{def}}{=} \text{Cat}(\mathbf{y}_n|\text{softmax}(W_\phi \mathbf{h}_{\phi, n})), \quad \text{with} \quad \mathbf{h}_{\phi, n} = g(\bar{\mathbf{y}}_{\text{NB}(n)}, E), \end{aligned}$$

where the object representation $\mathbf{h}_{\phi, n}$ is learned by feeding $\{\bar{\mathbf{y}}_{n'}\}_{n' \in \text{NB}(n)}$ as features in a graph neural network g .

Comparison. We empirically compare different methods in the semi-supervised object classification task, where we use 10 samples for the multi-sample method and the parameter τ in the annealing method is set as 0.1. Tab. 10 presents the results. We see the annealing method consistently outperforms other methods on all datasets, and therefore we use the annealing method for all the experiments in the paper.

B.5. Best Results with Standard Deviation

Table 11. Best results on semi-supervised object classification (%).

| Algorithm | Cora | Citeseer | Pubmed |
|------------|--------------------|--------------------|--------------------|
| GAT | 83.0 \pm 0.7 | 72.5 \pm 0.7 | 79.0 \pm 0.3 |
| GMNN | 83.675 \pm 0.900 | 73.576 \pm 0.795 | 81.922 \pm 0.529 |
| p -value | < 0.0001 | < 0.0001 | < 0.0001 |

Finally, we present the best mean accuracy together with the standard deviation of GMNN over 100 runs in Tab. 11. The improvement over GAT is statistically significant.