A. The Optimality Condition for q_θ

Theorem A.1. *For the mean-field variational distribution, the optimal $q_\theta(\mathbf{y}_n|\mathbf{x}_V)$ of each node n is given by the following fixed-point condition:*

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

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_θ is to minimize the KL divergence between $q_\theta(\mathbf{y}_U)$ and $p_\phi(\mathbf{y}_U|\mathbf{y}_L)$. Based on that, if we consider each individual node n_0, the objective function for $q_\theta(\mathbf{y}_{n_0})$ is given as follows:

$$\begin{aligned}
&O(q_\theta(\mathbf{y}_{n_0})) = -\mathrm{KL}(q_\theta(\mathbf{y}_U)||p_\phi(\mathbf{y}_U|\mathbf{y}_L)) \\
&= \sum_{\mathbf{y}_U} q_\theta(\mathbf{y}_U)[\log p_\phi(\mathbf{y}_U|\mathbf{y}_L) - \log q_\theta(\mathbf{y}_U)] \\
&= \sum_{\mathbf{y}_U} \prod_n q_\theta(\mathbf{y}_n) \left[\log p_\phi(\mathbf{y}_U, \mathbf{y}_L) - \sum_n \log q_\theta(\mathbf{y}_n)\right] \\
&\quad + \text{const} \\
&= \sum_{\mathbf{y}_{n_0}} \sum_{\mathbf{y}_{U\setminus n_0}} q_\theta(\mathbf{y}_{n_0}) \prod_{n\neq n_0} q_\theta(\mathbf{y}_n) \\
&\quad \left[\log p_\phi(\mathbf{y}_U, \mathbf{y}_L) - \sum_n \log q_\theta(\mathbf{y}_n)\right] + \text{const} \\
&= \sum_{\mathbf{y}_{n_0}} q_\theta(\mathbf{y}_{n_0}) \sum_{\mathbf{y}_{U\setminus n_0}} \prod_{n\neq n_0} q_\theta(\mathbf{y}_n) \log p_\phi(\mathbf{y}_U, \mathbf{y}_L) \\
&- \sum_{\mathbf{y}_{n_0}} q_\theta(\mathbf{y}_{n_0}) \sum_{\mathbf{y}_{U\setminus n_0}} \prod_{n\neq n_0} q_\theta(\mathbf{y}_n) \\
&\quad \left[\sum_{n\neq n_0} \log q_\theta(\mathbf{y}_n) + \log q_\theta(\mathbf{y}_{n_0})\right] + \text{const} \\
&= \sum_{\mathbf{y}_{n_0}} q_\theta(\mathbf{y}_{n_0}) \log F(\mathbf{y}_{n_0}) - \sum_{\mathbf{y}_{n_0}} q_\theta(\mathbf{y}_{n_0}) \log q_\theta(\mathbf{y}_{n_0}) \\
&\quad + \text{const} \\
&= -\mathrm{KL}\left(q_\theta(\mathbf{y}_{n_0})||\frac{F(\mathbf{y}_{n_0})}{Z}\right) + \text{const}.
\end{aligned} \tag{1}$$

Here, Z is a normalization term, which makes $F(\mathbf{y}_{n_0})$ a valid distribution on $\mathbf{y}_{n_0}$, and we have

$$\begin{aligned}
\log F(\mathbf{y}_{n_0}) &= \sum_{\mathbf{y}_{U\setminus n_0}} \prod_{n\neq n_0} q_\theta(\mathbf{y}_n) \log p_\phi(\mathbf{y}_U, \mathbf{y}_L) \\
&= \mathbb{E}_{q_\theta(\mathbf{y}_{U\setminus n_0})}[\log p_\phi(\mathbf{y}_U, \mathbf{y}_L)].
\end{aligned}$$

Based on the Eq.(1), the optimal $q_\theta(\mathbf{y}_{n_0})$ is achieved when it equals to $\frac{F(\mathbf{y}_{n_0})}{Z}$, and thus we have:

$$\begin{aligned}
&\log q_\theta(\mathbf{y}_{n_0}) = \log F(\mathbf{y}_{n_0}) + \text{const} \\
&= \mathbb{E}_{q_\theta(\mathbf{y}_{U\setminus n_0})}[\log p_\phi(\mathbf{y}_U, \mathbf{y}_L)] + \text{const} \\
&= \mathbb{E}_{q_\theta(\mathbf{y}_{U\setminus n_0})}\left[\log p_\phi(\mathbf{y}_{n_0}|\mathbf{y}_{V\setminus n_0})\right] + \text{const} \\
&= \mathbb{E}_{q_\theta(\mathbf{y}_{U\setminus n_0})}\left[\log p_\phi(\mathbf{y}_{n_0}|\mathbf{y}_{\mathrm{NB}(n_0)})\right] + \text{const} \\
&= \mathbb{E}_{q_\theta(\mathbf{y}_{\mathrm{NB}(n_0)\cap U})}\left[\log p_\phi(\mathbf{y}_{n_0}|\mathbf{y}_{\mathrm{NB}(n_0)})\right] + \text{const}.
\end{aligned}$$

Here, $p_\phi(\mathbf{y}_{n_0}|\mathbf{y}_{V\setminus n_0}) = p_\phi(\mathbf{y}_{n_0}|\mathbf{y}_{\mathrm{NB}(n_0)})$ is based on the conditional independence property of Markov networks. □

B. Additional Experiment

B.1. Results on Random Data Splits

Table 1. 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**

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 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. 1. We see GMNN consistently outperforms GCN and GAT on all datasets, proving the effectiveness and robustness of GMNN.

B.2. Results on Few-shot Learning Settings

Table 2. 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-short 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. 2. 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 observation proves the effectiveness of GMNN even when labeled objects are extremely limited.

B.3. Comparison with Self-training Methods

Table 3. 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 seeks to annotate unlabeled objects, and the annotated objects are further treated as extra data to update q_θ through Eq. (**??**). Similarly, in self-training frameworks, we typically use q_θ itself to annotate unlabeled objects, and collect additional training data for q_θ. Next, we compare GMNN with the self-training method in the semi-supervised object classification task, and the results are presented in Tab. 3.

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

Table 4. 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

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. (**??**). Learning the optimal q_θ requires computing the right side of Eq. (**??**), 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 $q_\theta(\mathbf{y}_{\text{NB}(n)\cap U}|\mathbf{x}_V)$ can be factorized as $\prod_{k\in\text{NB}(n)\cap U} q_\theta(\mathbf{y}_k|\mathbf{x}_V)$. Based on that, we can develop several approximation methods.

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

Multiple Samples. In practice, we can also draw multiple samples from $q_\theta(\mathbf{y}_k|\mathbf{x}_V)$ 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 $q_\theta(\mathbf{y}_k|\mathbf{x}_V)$, such that we have:

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

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

Max Pooling. Another method is max pooling, where we set $\hat{\mathbf{y}}_k = \arg\max_{\mathbf{y}_k} q_\theta(\mathbf{y}_k|\mathbf{x}_V)$ for $k \in \text{NB}(n) \cap U$, and use $\{\hat{\mathbf{y}}_k\}_{k\in\text{NB}(n)\cap U}$ 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, we set $\bar{\mathbf{y}}_k = \mathbb{E}_{q_\theta(\mathbf{y}_k|\mathbf{x}_V)}[\mathbf{y}_k]$ for each unlabeled neighbor k of the object n, which can be understood as a soft label vector of that neighbor. For each labeled neighbor k of the object n, we set $\bar{\mathbf{y}}_k$ as the one-hot label vector. Then we can approximate the expectation as:

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

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

$$\mathbf{h}_{\phi,n} = g(\bar{\mathbf{y}}_{\text{NB}(n)}, E).$$

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. 4 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 5. Best results on semi-supervised object classification (%).

Algorithm	Cora	Citeseer	Pubmed
GAT	83.0 ± 0.7	72.5 ± 0.7	79.0 ± 0.3
GMNN	83.675 ± 0.900	73.576 ± 0.795	81.922 ± 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. 5. The improvement over GAT is statistically significant.

References

Deng, Y., Kim, Y., Chiu, J., Guo, D., and Rush, A. Latent alignment and variational attention. In *NeurIPS*, 2018.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *ICLR*, 2018.

Yang, Z., Cohen, W., and Salakhudinov, R. Revisiting semi-supervised learning with graph embeddings. In *ICML*, 2016.