Carnegie Mellon University

Approximate Inference with Graph Neural Networks

Lingxiao Zhao, Ksenia Korovina, Wenwen Si, Mark Cheung

P21

Introduction

- Graphical models provide a helpful bridge between probability and graphs, transforming a probabilistic inference task into operations on a graph. However, exact inference in graphical models is NP-hard.
- Approximate inference algorithms are widely used to tackle the problem of exact inference, such as Loopy Belief Propagation (LBP), MCMC, and Variational Inference (VI).
- Graph Neural Network (GNN) becomes widely used for solving problems over graphs and achieves great performance over many tasks. **Could GNN be applied to solve inference over PGM?** We follow the work of Yoon et al. [2018] in investigating the ability of GNN for approximate inference.

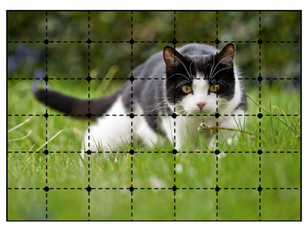
Overview

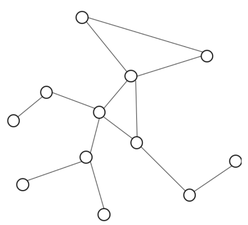
Shortcoming of Existing Approximate Inference

- **(L)BP**: works great for tree but bad for non-tree graphs.
- **MCMC**: needs a lot of computational resources.

Graph Neural Networks (GNN):

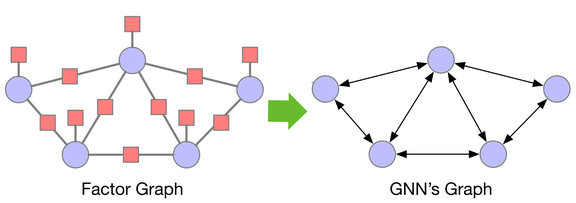
- GNNs are motivated by CNNs, which can extract meaningful features for Euclidean data (e.g., images). GNNs are CNNs applied to non-Euclidean domains.





Mapping inference to GNN:

- Variable maps to node, factor information becomes edge's input inside each layer of GNN:



Proposed Method

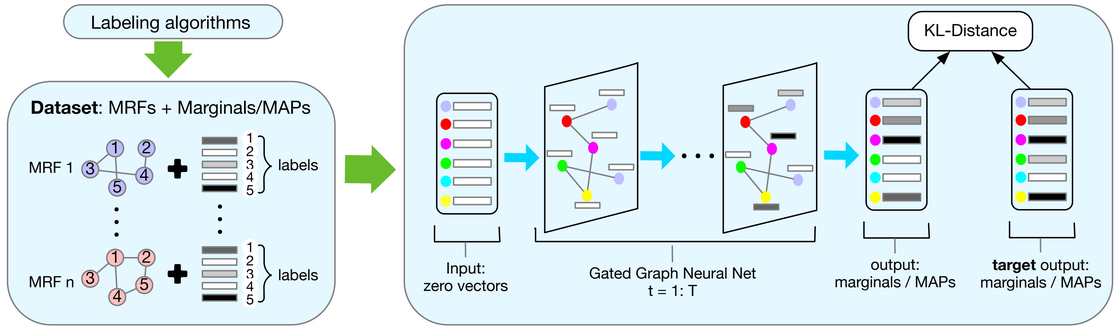


Figure: Pipeline for the proposed GNN-based method: GNN inference

Labeling stage

- In experiments, we use binary MRFs with variables $\boldsymbol{x} \in \{-1, +1\}^{|V|}$ and parameters $\boldsymbol{b}$ and a symmetric matrix $\boldsymbol{J}$ with the probability:

$$p(\boldsymbol{x}) = Z^{-1} \exp\left(\boldsymbol{b}\boldsymbol{x} + \boldsymbol{x}^T \boldsymbol{J} \boldsymbol{x}\right)$$

- Graphs sampled i.i.d. $u_{i,j} \sim N(0,1), b_i \sim N(0, (\frac{1}{4})^2)$, $W = (U + U^T)/2$.
- Training labels: we can generate either exact labels (intractable!) or approximate labelers
- Testing labels: exact, BP (trees), MCMC (non-trees)

Training+inference stage

- At each step t, update the edge messages: $\boldsymbol{m}_{i\to j}^{t+1} = MLP(\boldsymbol{h}_i^t, \boldsymbol{h}_j^t, \boldsymbol{e}_{ij})$, where $\boldsymbol{e}_{ij} = [J_{i,j}, b_i, b_j]$.
- Neighborhood information is calculated by aggregating all incoming messages: $\boldsymbol{m}_i^{t+1} = \sum_{j \in N_i} \boldsymbol{m}_{j\to i}^{t+1}$
- Hidden state update: $\boldsymbol{h}_i^{t+1} = GRU(\boldsymbol{h}_i^t, \boldsymbol{m}_i^{t+1})$
- After T iterations, readout: $\hat{\boldsymbol{y}}_i = \text{softmax}(MLP(\boldsymbol{h}_i^T))$
- Training loss: KL-divergence
- To perform marginal inference on G, return GNN(G). For MAP inference, use a threshold.

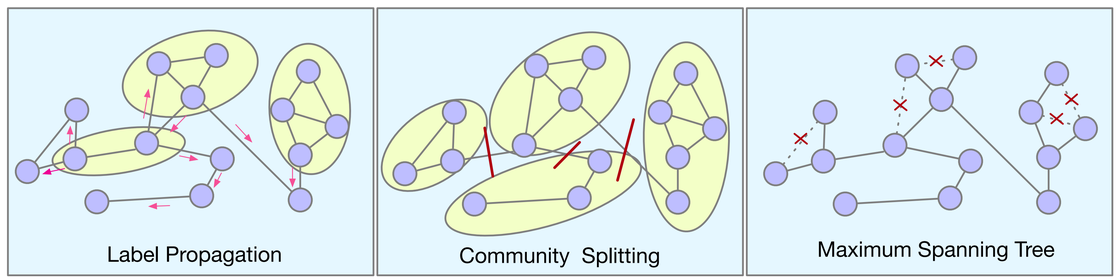


Figure: Proposed algorithms for approximate labeling for GNN training.

Label prop

We use our variant Label propagation:

1. Label several random subgraphs with a "reliable" algorithm (e.g. exact).
2. Propagate information to new nodes for T steps:

$$\hat{p}_j(+1) \leftarrow \sigma\left(\sum_j w_{j,i} \hat{p}_j(\text{sign}(w_{j,i}))\right)$$

Community splitting

1. Split the graph into manageable subgraphs that are least connected.
2. Algorithms: Girvan-Newman, Louvain, other options
3. Label each community separately.
4. (ongoing) Merge community based on their probabilities, and biases b

MST

1. Find the maximum absolute weight spanning tree:

$$T = \arg\max_T \sum_{(i,j)\in T} |w_{i,j}|$$

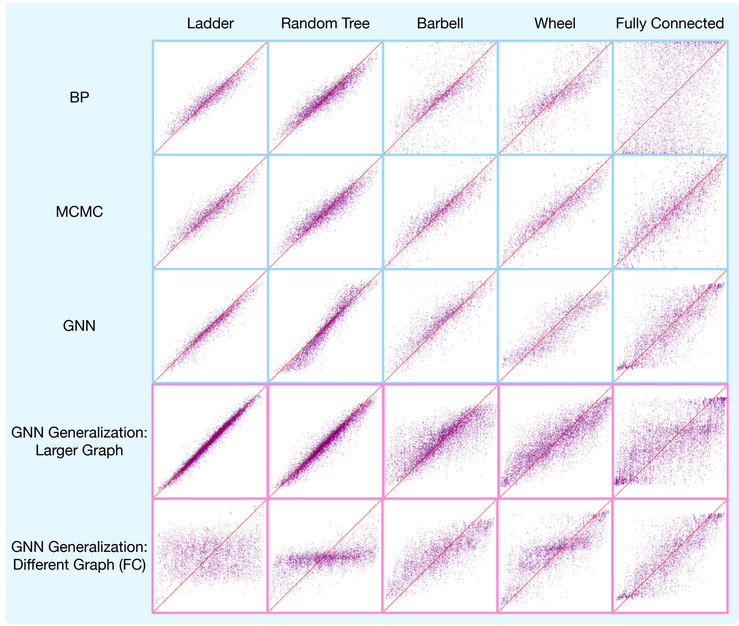
2. Use exact belief propagation on tree.

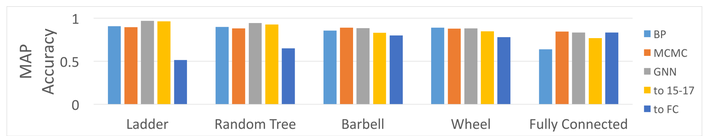
Future Work

- Modifying and tuning approximate labeling algorithms to improve their performance
- Applying GNN as inference over real-world problem, such as image segmentation.

Suite 1: GNN Generalization

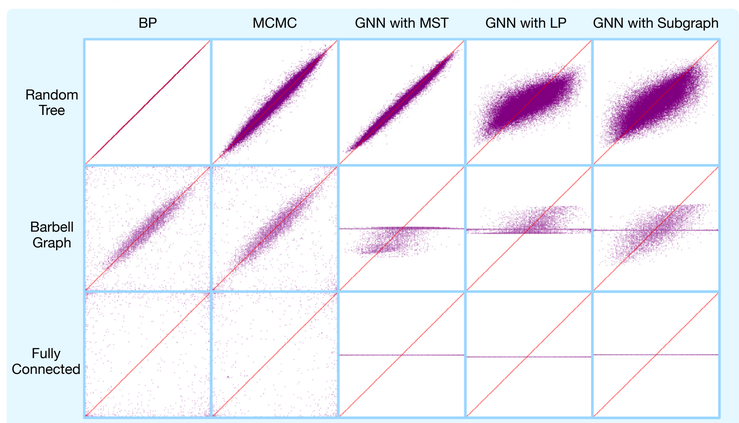
Figure: Comparison of different algorithms for in-sample and out-sample generalization





Suite 2: GNN Scalability (Preliminary)

Figure: Comparison of different labeling methods for graph of size 100



Graphs of size 15-17 (sec/graph)

	sparse GGNN	BP	MCMC
Path/star	0.0098	**0.0068**	0.2797
Bipart/FC	**0.0132**	0.1537	0.3032

Graphs of size 100 (sec/graph)

	sparse GGNN	BP	MCMC
Rand.tree	**0.0142**	0.0953	1.776
Barb+FC	**0.1605**	6.594	1.874

References

KiJung Yoon, Renjie Liao, Yuwen Xiong, Lisa Zhang, Ethan Fetaya, Raquel Urtasun, Richard S. Zemel, and Zachary Pitkow. Inference in probabilistic graphical models by graph neural networks. *CoRR*, abs/1803.07710, 2018.