IBM Research

Mattson Thieme[1,2], Han Liu[1] (Advisor), Yada Zhu[1] (Mentor)

[1] MIT-IBM Watson AI Lab, Exploratory AI Research
[2] Northwestern University

Learnable Sparsification without Constraints

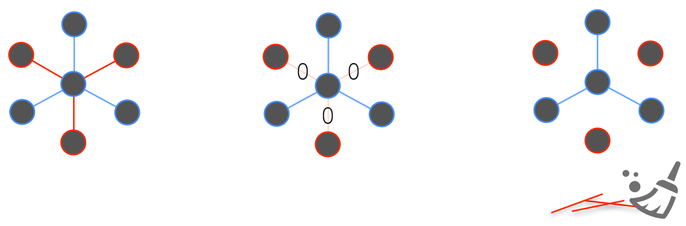
Motivation

Most graphs are constructed using heuristics and, as such, contain noisy edges. Below, we show the performance of a Graph Attention Network (GAT) on a noised version of the Cora dataset, where we have added three random edges per node.

The first cell shows the default GAT's performance on this dataset. The second shows the performance of the GAT if we manually set the edge weights of the noisy edges to zero (note: this is the ideal case, the GAT does not learn this on its own).
The third shows the performance of the GAT if we *remove* the noisy edges altogether.

	Default GAT	Manually *zero* noisy edges	Manually *drop* noisy edges
Classification Accuracy	65.9%	79.0%	82.1%

Cora + Noise

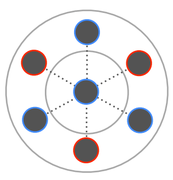


Challenge
Most graphs contain noisy edges, and GNN's are highly sensitive to network structure. In many cases, aggregating neighborhood information in a GNN actually harms prediction performance.

Node Embedding conflicts with Structure Learning

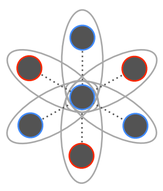
Node Embedding: Relative Edge Weighting

- The node embedding mechanism must be able to account for neighborhoods of varying size. As such, it must normalize each neighbor with respect to all the others.



Structure Learning: Absolute Edge Weighting

- Learning discrete structures subjects the graph to noisy evolution, during which time we must assess the value of each candidate neighbor independent of the present network configuration.



New Idea
Dropping noisy edges is better than zeroing their weights, and the mechanism for dropping those edges should be invariant with respect to the network structure.

The Graph Learning Attention Mechanism (GLAM)

Like the typical GAT, the input to our layer is $\mathbf{h} = \{\overrightarrow{h_1}, \cdots, \overrightarrow{h_N}\}, \overrightarrow{h_i} \in \mathbb{R}^F$ where N is the number of nodes and F is the number of features in each node. Unlike GAT, GLAM outputs a binary mask $\mathbf{M} \in \{0,1\}^{|\mathscr{E}|}$ where $|\mathscr{E}|$ is the cardinality of the edge set. Given the input $\mathbf{h}$, our model yields structure learning scores η which represent, for each edge, the probability that we retain that edge:

$$\eta = \sigma\Big(\mathbf{S}\big[\mathsf{ELU}\big(\mathbf{W}\overrightarrow{h_i}\big) \,\|\, \mathsf{ELU}\big(\mathbf{W}\overrightarrow{h_j}\big)\big]\Big), \eta \in \mathbb{R}^{|\mathscr{E}|\times 2}$$

Where $\mathbf{W} \in \mathbb{R}^{F_s \times F}$ is a shared linear transformation mapping node features into higher level representations, $\|$ is vector concatenation, $\mathbf{S} \in \mathbb{R}^{2\times F_s}$ is another shared linear transformation mapping edge representations into *structure learning scores*, and σ is a softmax mapping each set of structure learning scores $\in \mathbb{R}^{1\times 2}$ into probabilities $[P(\text{discard}), P(\text{retain})]$.

Finally, we sample a binary mask $M \in \{0,1\}^{|\mathscr{E}|}$ from this distribution using the Gumbel Softmax Reparameterization Trick, which yields a new edge set $M(\mathscr{E}) \rightarrow \mathscr{E}' \in \mathscr{E}$ where $|\mathscr{E}'| \leq |\mathscr{E}|$. $\mathscr{E}'$ then defines the computation graph for the downstream GNN representation learner. If the downstream GNN is a GAT, its final attention weights α_{ij} would then become:

$$\alpha_{ij} = \mathsf{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k\in\mathscr{N}'_i}\exp(e_{ij})}$$

Which we show here in order to demonstrate the operational change: for each node, the downstream GNN attends over its *learned* neighborhood $\mathscr{N}'_i$, taken from the learned edge set $\mathscr{E}'$, and *not* over its given neighbors $\mathscr{N}_i$ in the given edge set $\mathscr{E}$. Here, the values e_{ij} are attention coefficients internal to the GAT layer.

Structure Learning without heuristics

Our formulation does not rely on any exogenous heuristics for structure learning, such as top-k selection or penalties for retained edges. The only training signal is from the task performance.

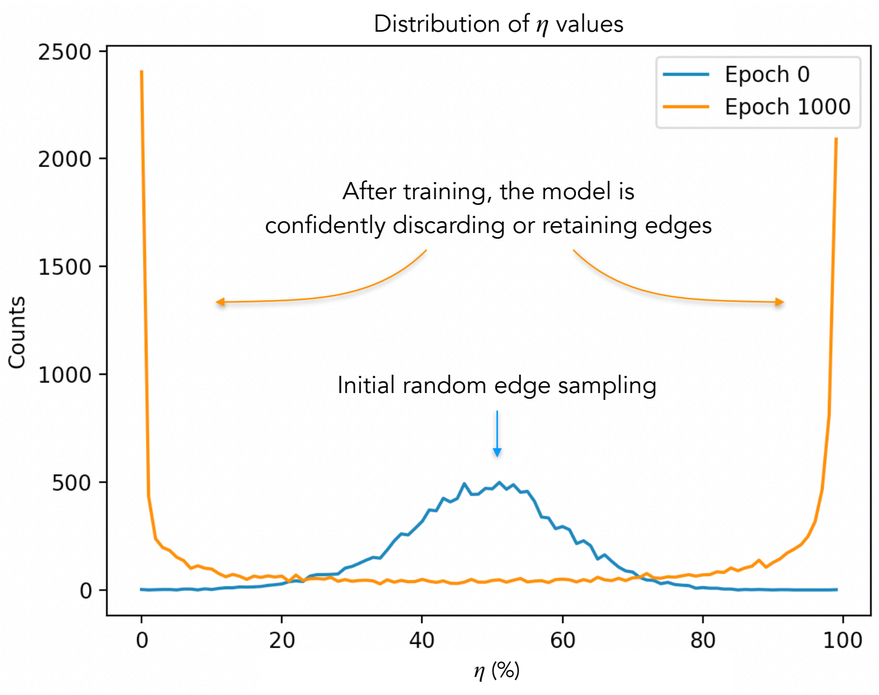
Results

Our model is able to match or beat SOTA on a variety of node prediction tasks. Here, we review performance on both homophilic and heterophilic graphs, where neighborhood aggregation is known to be either helpful or harmful, respectively.

	Homophilic			Heterophilic		
	Cora	Citeseer	PubMed	Cornell	Texas	Wisconsin
GAT	**82.1%**	69.7%	77.3%	49.2%	45.4%	48.2%
GLAM + GAT	**82.1%**	**70.6%**	**77.5%**	**73.0%**	**71.9%**	**81.2%**
Dropped %	2.7%	7.0%	1.0%	100%	99.0%	99.0%
	Our model recognizes the utility in aggregating neighborhood information, keeps most of the edges and performs on-par with the baseline models.			*Our model recognizes the negative utility of aggregating neighborhood information and effectively converts itself into an MLP by dropping almost every edge.*		
	Edges in homophilic graphs tend to join similar nodes, allowing neighborhood aggregation to increase performance.			Edges in heterophilic graphs tend to join dissimilar nodes, degrading the performance of graph neural networks. This effect is so strong that MLPs routinely outperform GNNs on node prediction in heterophilic graphs.		

Note: some results are still 0.5-1% below reported SOTA. Optimization is ongoing, but we have verified that the additional performance gain realized by GLAM is not due to it simply acting as a dropout regularizer.

Training dynamics are smooth and behave as expected



As training proceeds, the model unambiguously discards/retains the edges, with edge probabilities approaching either 0 or 1. Note: edge sampling during validation/testing is deterministic.

Results
Our model is able to recognize and drop edges that decrease performance, allowing the downstream GNN to match or exceed SOTA performance on benchmark datasets.

Contact Information: Mattson Thieme, mattson.thieme@gmail.com, (971) 341-7808