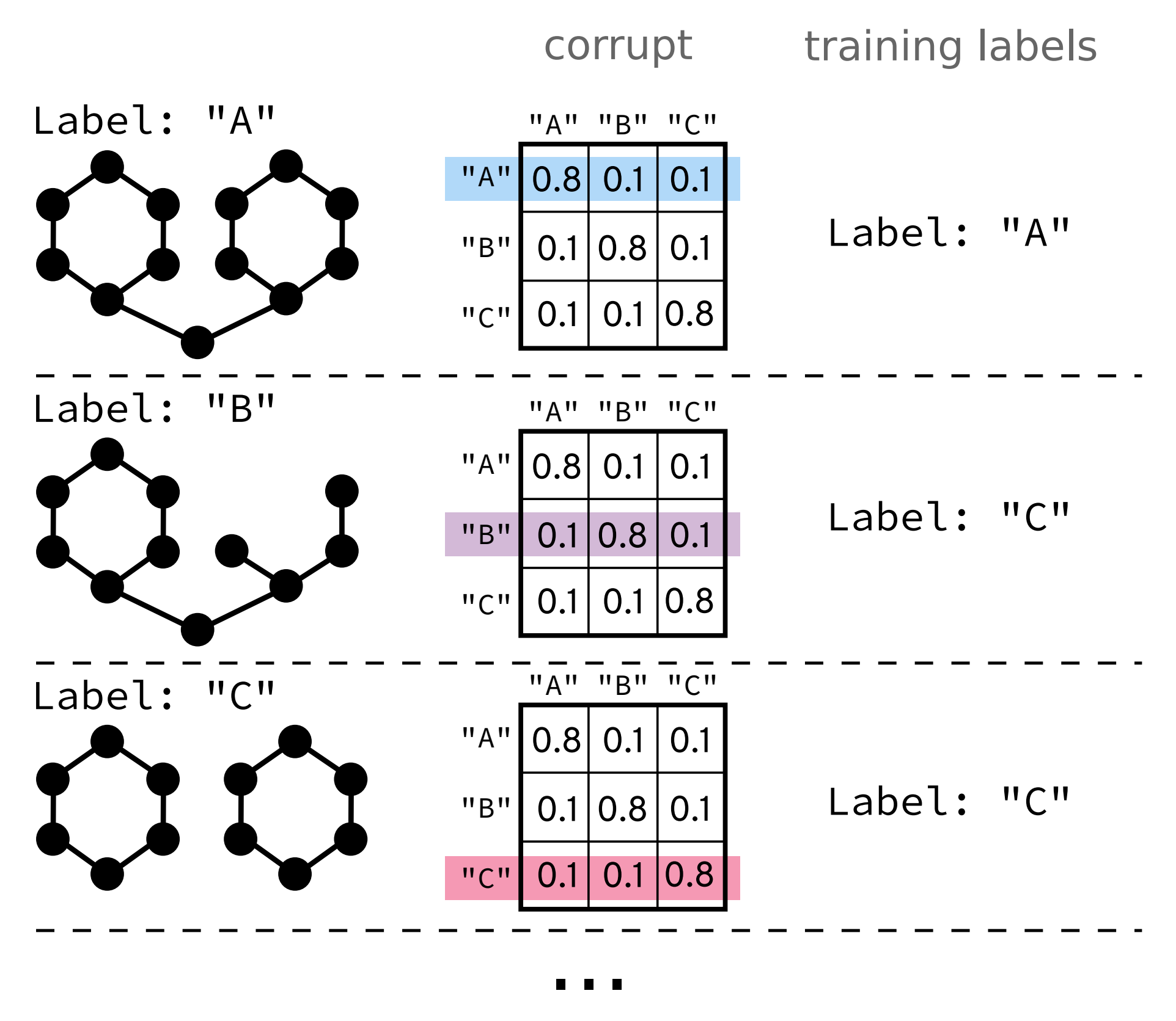
Learning GNNs with Noisy Labels

Hoang NT, Jun Jin Choong, Tsuyoshi Murata
Tokyo Institute of Technology

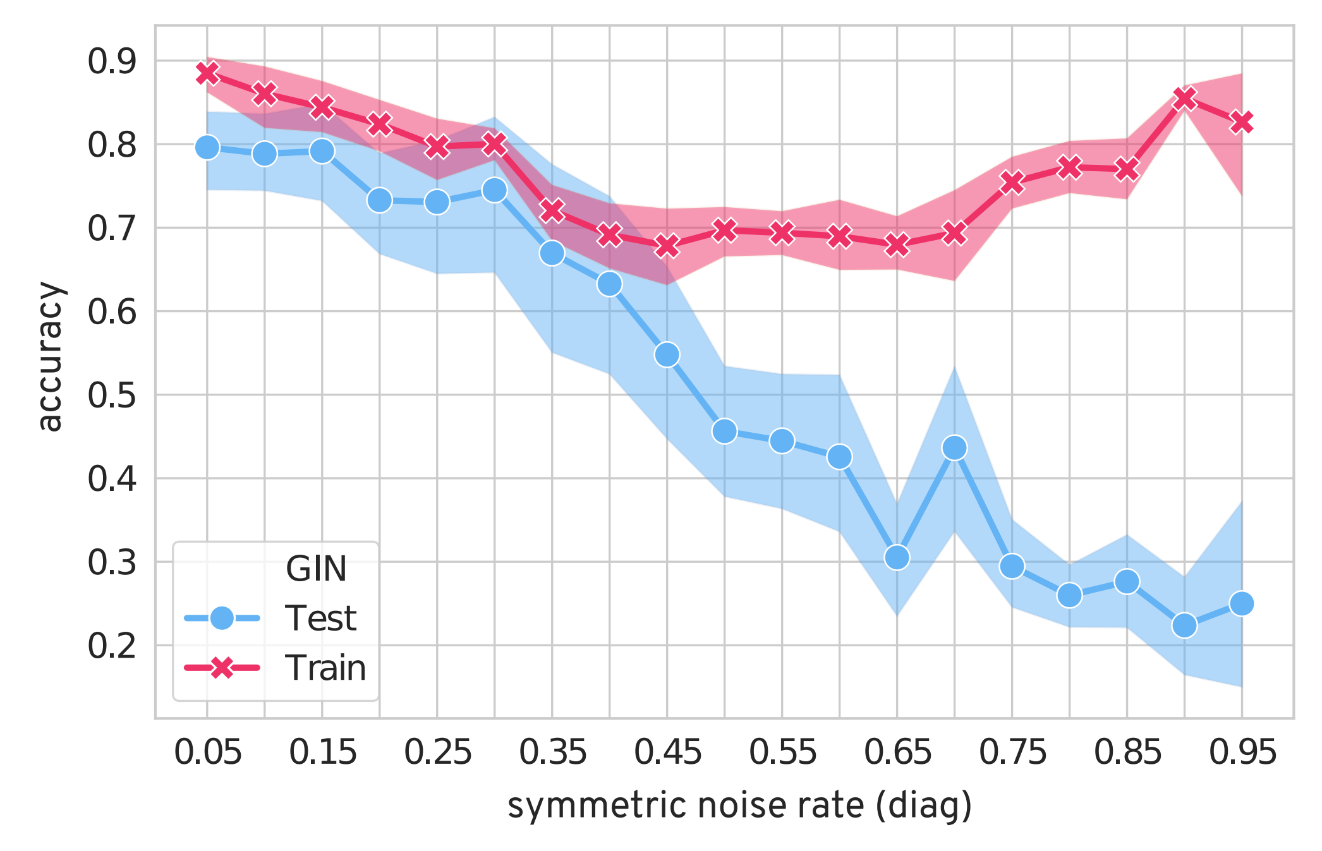
東京工業大学 Tokyo Institute of Technology
LEARNING WITH LIMITED LABELED DATA
ICLR 2019 New Orleans, USA

Introduction

We study the robustness to symmetric label noise of GNNs training procedures. By combining the nonlinear neural message-passing models (e.g. Graph Isomorphism Networks, GraphSAGE, etc.) with loss correction methods, we present a noise-tolerant approach for the graph classification task. Our experiments show that test accuracy can be improved under the artificial symmetric noisy setting.



MUTAG GENERALIZATION UNDER FOR NOISY LABELS



Graph Classification Model

We use a nonlinear message passing model similar to *GIN* [2] and *GraphSAGE* [3]. In the supervised learning setting, such model has two main steps:

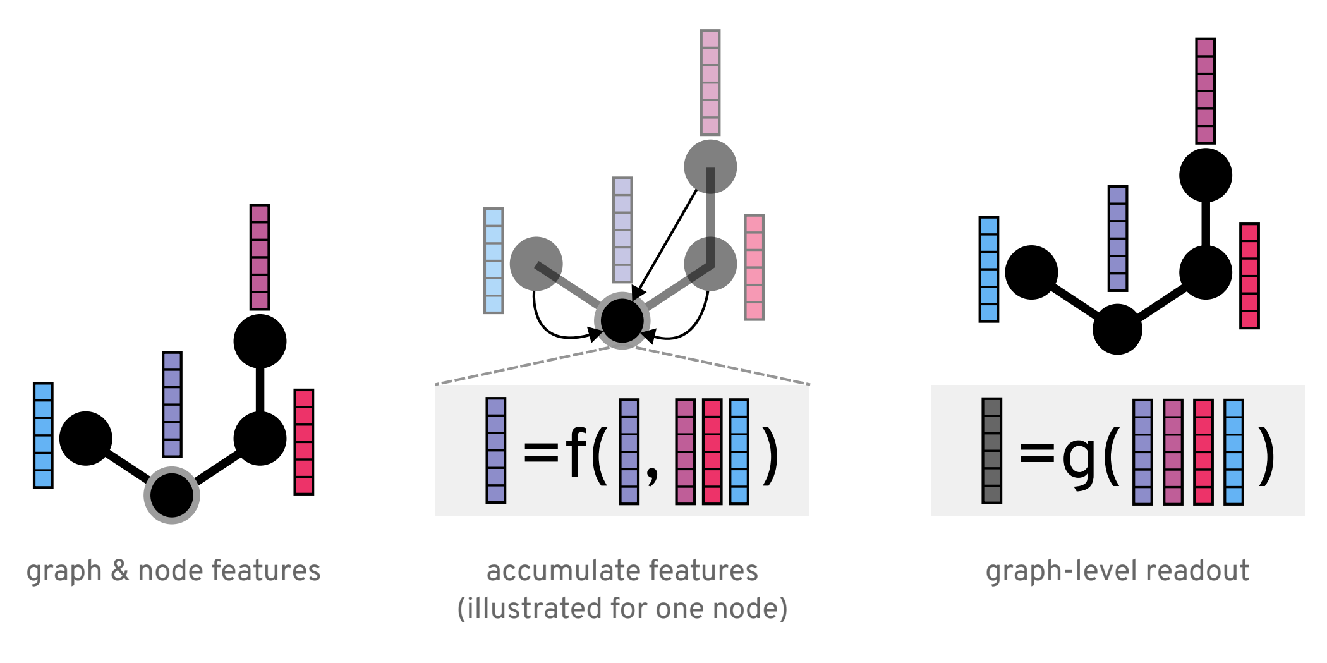
1. Accumulate neighborhood information to each vertex $\boldsymbol{v}$ and neural network layer (k):

$$\mathbf{a}_v^{(k)} = \texttt{AGGREGATE}^{(k)}(\{\mathbf{h}_u^{(k-1)} : u \in \mathcal{N}(v)\}),$$
$$\mathbf{h}_v^{(k)} = \texttt{COMBINE}^{(k)}(\mathbf{h}_v^{(k-1)}, \mathbf{a}_v^{(k)})$$

2. Employ a function to learn the overall representation of the graph, then the objective ℓ is optimized with standard backpropagation.

$$\mathbf{h}_{\mathcal{G}} = \texttt{READOUT}(\{\mathbf{h}_v^{(K)} : v \in \mathcal{G}\}),$$
$$\ell(p(y|\mathbf{h}_{\mathcal{G}}), y_{\mathcal{G}}) = \texttt{XENTROPY}(p(y|\mathbf{h}_{\mathcal{G}}), y_{\mathcal{G}})$$

$\mathbf{h}_v^{(k)}$: Feature vector of node v at iteration (k); $\mathbf{h}_{\mathcal{G}}$: Feature vector of whole graph; $y_{\mathcal{G}}$: True label of training graphs; $p(y|\mathbf{h}_{\mathcal{G}})$: A prediction model.



Short References

[1] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Q. "Making deep neural networks robust to label noise: A loss correction approach," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1944–1952, 2017.
[2] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. "How powerful are graph neural networks?," *International Conference on Learning Representations*, ICLR 2019.
[3] Will Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs," *Advances in Neural Information Processing Systems*, pp. 1024–1034, 2017.
[4] Thomas N. Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks," *International Conference on Learning Representations*, ICLR 2017.

Loss Correction Schemes

We employ the loss correction technique from the line of work in [3]. We propose to use the *backward* surrogate loss with different estimators C for the noise matrix N:

$$\ell^{\leftarrow} = \boldsymbol{C}^{-1} \cdot \ell(\hat{p}(y|\mathcal{G}))$$

Our final loss function for the GNN model is:

$$\ell^{\leftarrow}(p(y|\mathbf{h}_{\mathcal{G}}), y_{\mathcal{G}}) = \boldsymbol{C}^{-1} \cdot \texttt{CROSS\_ENTROPY}(p(y|\mathbf{h}_{\mathcal{G}}), y_{\mathcal{G}})$$

Such loss correction can be understood as going backward one step in the noisy process. To estimate $\boldsymbol{C}$ from the noisy data (corrupted by some unknown noise process $\boldsymbol{N}$), we propose two estimation schemes:

1. Assume the neural network can exactly model the noisy data, hence the final softmax output is used to fill matrix $\boldsymbol{C}$. For instance, out of all sample classified as class "A" in our training data, we find the one that gives the weakest response for class "A", and use its softmax output as the values for row "A" in matrix $\boldsymbol{C}$.
2. two

Dataset (#classes)	diag(N)	Avg. diag($\boldsymbol{C}^{\mathrm{c}}$)	$\lVert\boldsymbol{C}^{\mathrm{c}} - N\rVert$	Avg. diag($\boldsymbol{C}^{\mathrm{a}}$)	$\lVert\boldsymbol{C}^{\mathrm{a}} - N\rVert$
IMDB-B (2)	0.8	0.99	0.76	0.77	0.12
IMDB-M (3)	0.8	0.99	1.14	0.85	0.30
RDT-B (2)	0.8	0.99	0.76	0.75	0.20
RDT-M5K (5)	0.8	0.99	1.90	0.81	0.10
COLLAB (3)	0.8	0.99	1.14	0.75	0.30
MUTAG (2)	0.8	0.99	0.76	0.74	0.24
PROTEINS (2)	0.8	0.99	0.76	0.78	0.08
PTC (2)	0.8	0.99	0.76	0.63	0.68
NCI1 (2)	0.8	0.99	0.76	0.74	0.24

Experimental Results

Dataset	#graphs	#classes	#vertices
IMDB-B	1000	2	19.8
IMDB-M	1500	3	13.0
RDT-B	2000	2	429.6
RDT-M5K	5000	5	508.5
COLLAB	5000	3	74.5
MUTAG	188	2	17.9
PROTEINS	1113	2	39.1
PTC	344	2	25.5
NCI1	4110	2	29.8

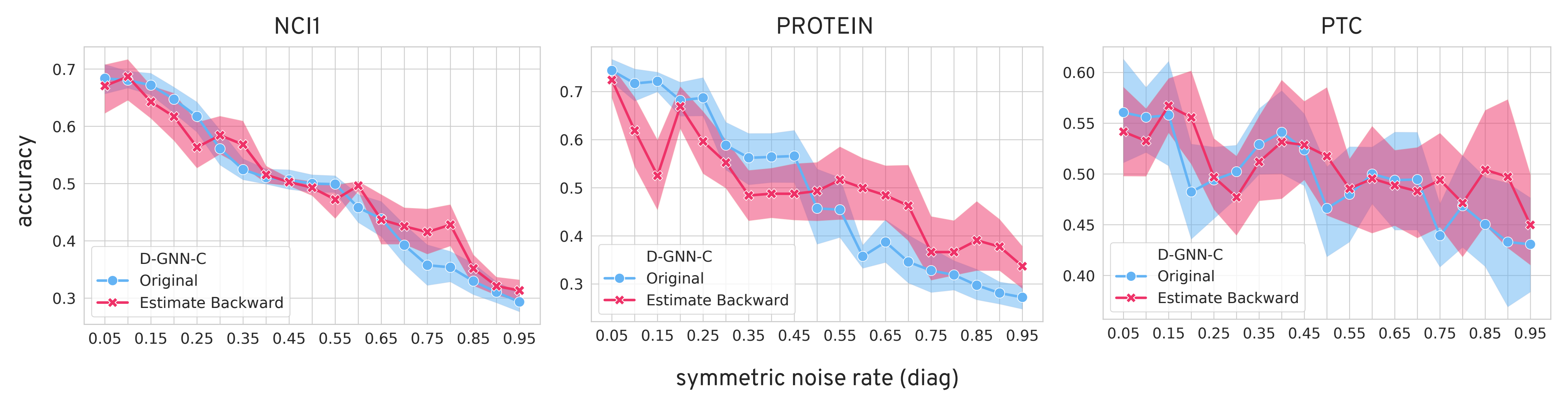
In this experiment, we simulate a robot keeping station, repeatedly scanning the same area.

Top: Our method, BGKOctoMap updated with information from 1, 15, 30, and 60 scans (**left** to **right**) containing the same data.

Bottom: The result after applying the online Gaussian process occupancy mapping method in [?] to the same data.

Qualitatively, our method (**top**) is more stable for long-term mapping scenarios with many repeat observations than the previous method (**bottom**) in which the occupied voxels tend to grow continuously with repeat observations.

	MUTAG	IMDB-M	RDT-B	RDT-M5K	COLLAB	IMDB-B	PROTEINS	PTC	NCI1
GIN	.7327	.4476	.6695	.3677	.6544	.6573	.6257	.4824	.6472
GraphSAGE	.7072	.4373	-	-	-	.6410	.6583	.4892	.6053
D-GNN-C	.5727	**.4747**	.5005	.2000	.5979	**.6940**	**.6693**	**.5557**	.6170
D-GNN-A	.7102	**.4505**	.5307	.2000	**.6917**	**.7088**	**.6769**	**.5001**	.6405
D-GNN-E	.7002	**.4633**	.5270	.2022	**.6960**	**.7190**	**.6917**	**.5235**	**.6638**



We also evaluated our method qualitatively on real data from the University of Freiburg, shown above. From **left** to **right** we have:
1) The raw range-sensor data from University of Freiburg Corridor FR-079
2) The map produced by standard OctoMap
3) The map produced by BGKOctoMap

The source code is available at: `github.com/gear/denoising-gnn`.