Graph Structure Learning For GNNs

Learn the optimal graph for GNNs

Xuan Kan
xuan.kan@emory.edu
Department of Computer Science, Emory University

Learning Discrete Structures for Graph Neural Networks

Luca Franceschi, Mathias Niepert, Massimiliano Pontil, Xiao He. ICML 2019

MOTIVATION

1. Graph neural networks (GNNs) are a popular and powerful since they can incorporate graph structures.

2. In practice, however, real-world graphs are often noisy and incomplete or might not be available at all.

TARGET

1. Firstly, learn a graph structure through datapoints or complete the incomplete graph.
2. Then apply GNNs with the new graph to achieve better performance.

PRELIMINARY: GCN

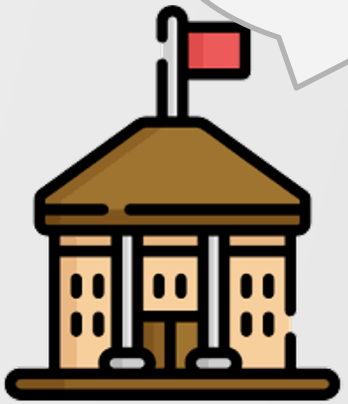
$$f_w(X, A) = \mathrm{Softmax}(\hat{A}\ \mathrm{ReLu}(\hat{A}\ X\ W_1)\ W_2),$$

$$\hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$$

PRELIMINARY: Bilevel Programming

Toll-setting problem

How to set tolls can maximize my revenues.



Outer Problem

After toll setting, how to choose routes can minimize my traveling costs.

Inner Problem

To find the optimal toll setting, government need to consider drivers' options

METHOD: Define the problem

Outer Problem: Find a graph, minimize the validation loss.

θ, ω are learnable parameters.

$$\min_{\theta \in \overline{\mathcal{H}}_N} \mathbb{E}_{A \sim \mathrm{Ber}(\theta)} \left[F(w_\theta, A) \right]$$
$$\text{such that } \; w_\theta = \arg\min_w \mathbb{E}_{A \sim \mathrm{Ber}(\theta)} \left[L(w, A) \right].$$

Inner Problem: Given a graph, learn a GNN to minimize the training loss.

METHOD: Optimize the problem

1. Treat the graph structure as hyperparameters. (Meta learning problem)
2. Common methods to learn optimal hyperparameters:
    1. Grid Search
    2. Random Search
    3. Bayesian Optimization
    4. Reinforcement Learning
    5. Gradient-based Methods

METHOD: Overall Flow

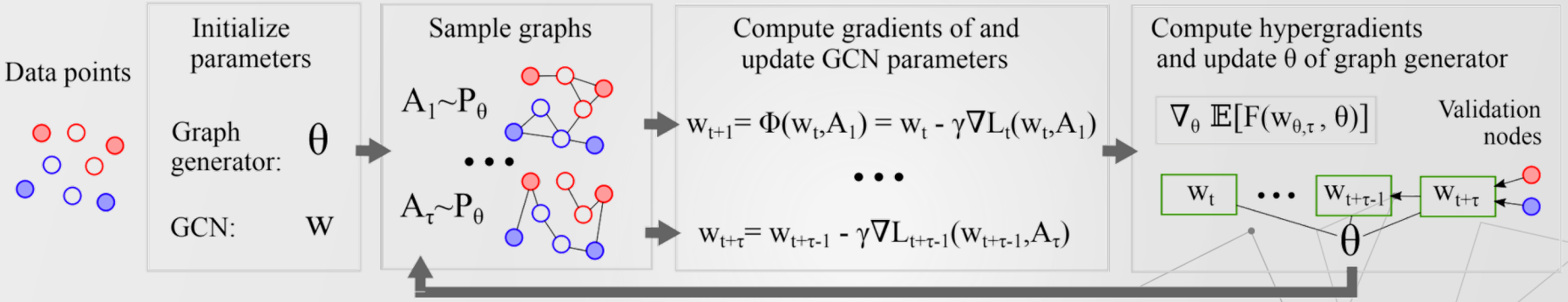


Figure 1. Schematic representation of our approach for learning discrete graph structures for GNNs.

METHOD: Graph Generator

1. Learnable parameters: $\theta \in \mathbb{R}^{n*n}$, $\theta_{i,j}$ means the probability that node i connects to node j.
2. Input: Nothing
3. Output: adjacency matrix A, $A \sim Ber(\theta)$.

Method: Algorithm

Algorithm 1 LDS

Input data: $X, Y, Y'[, A]$
Input parameters: $\eta, \tau[, k]$
$[A \leftarrow \texttt{kNN}(X, k)]$ {Init. A to kNN graph if $A = 0$}
$\theta \leftarrow A$ {Initialize P_θ as a deterministic distribution}
while Stopping condition is not met **do**
  $t \leftarrow 0$
  while Inner objective decreases **do**
    $A_t \sim \text{Ber}(\theta)$ {Sample structure}
    $w_{\theta,t+1} \leftarrow \Phi_t(w_{\theta,t}, A_t)$ {Optimize inner objective}
    $t \leftarrow t + 1$
    if $t = 0 \pmod{\tau}$ **or** $\tau = 0$ **then**
      $G \leftarrow \texttt{computeHG}(F, Y, \theta, (w_{\theta,i})_{i=t-\tau}^{t})$
      $\theta \leftarrow \text{Proj}_{\overline{\mathcal{H}}_N}[\theta - \eta G]$ {Optimize outer objective}
    end if
  end while
end while
return w, P_θ {Best found weights and prob. distribution}

Method: Algorithm

Algorithm 1 LDS

Input data: $X, Y, Y'[, A]$
Input parameters: $\eta, \tau[, k]$
$[A \leftarrow \texttt{kNN}(X, k)]$ {Init. A to kNN graph if $A = 0$}
$\theta \leftarrow A$ {Initialize P_θ as a deterministic distribution}
while Stopping condition is not met **do**
  $t \leftarrow 0$
  while Inner objective decreases **do**
    $A_t \sim \text{Ber}(\theta)$ {Sample structure}
    $w_{\theta,t+1} \leftarrow \Phi_t(w_{\theta,t}, A_t)$ {Optimize inner objective}
    $t \leftarrow t + 1$
    if $t = 0 \pmod{\tau}$ **or** $\tau = 0$ **then**
      $G \leftarrow \texttt{computeHG}(F, Y, \theta, (w_{\theta,i})_{i=t-\tau}^{t})$
      $\theta \leftarrow \text{Proj}_{\overline{\mathcal{H}}_N}[\theta - \eta G]$ {Optimize outer objective}
    end if
  end while
end while
return w, P_θ {Best found weights and prob. distribution}

Method: Algorithm

Algorithm 1 LDS

Input data: $X, Y, Y'[, A]$
Input parameters: $\eta, \tau[, k]$
$[A \leftarrow \texttt{kNN}(X, k)]$ {Init. A to kNN graph if $A = 0$}
$\theta \leftarrow A$ {Initialize P_θ as a deterministic distribution}
while Stopping condition is not met **do**
  $t \leftarrow 0$
  while Inner objective decreases **do**
    $A_t \sim \text{Ber}(\theta)$ {Sample structure}
    $w_{\theta,t+1} \leftarrow \Phi_t(w_{\theta,t}, A_t)$ {Optimize inner objective}
    $t \leftarrow t + 1$
    if $t = 0 \pmod{\tau}$ **or** $\tau = 0$ **then**
      $G \leftarrow \texttt{computeHG}(F, Y, \theta, (w_{\theta,i})_{i=t-\tau}^{t})$
      $\theta \leftarrow \text{Proj}_{\overline{\mathcal{H}_N}}[\theta - \eta G]$ {Optimize outer objective}
    end if
  end while
end while
return w, P_θ {Best found weights and prob. distribution}

EXPERIMENTS

1. Node classification on incomplete graphs
    1. Aim: Evaluate the robustness and effectiveness with incomplete graphs.
    2. Method: Construct graphs with missing edges by randomly sampling 25%, 50%, and 75% of the edges on Cora and Citeseer.
2. Classification on dataset without graph structures
    1. Aim: Evaluate the effectiveness on dataset without graph structures.
    2. Method: Build a population graph among samples, then evaluate as the node classification task.
3. Visualize the probability distribution in $\boldsymbol{\theta}$
    1. Aim: Demonstrate why learnable graph can improve node classification performance.
    2. Method: Compare the average probability that a node connects to these same class nodes and other class nodes.

EXPERIMENT1

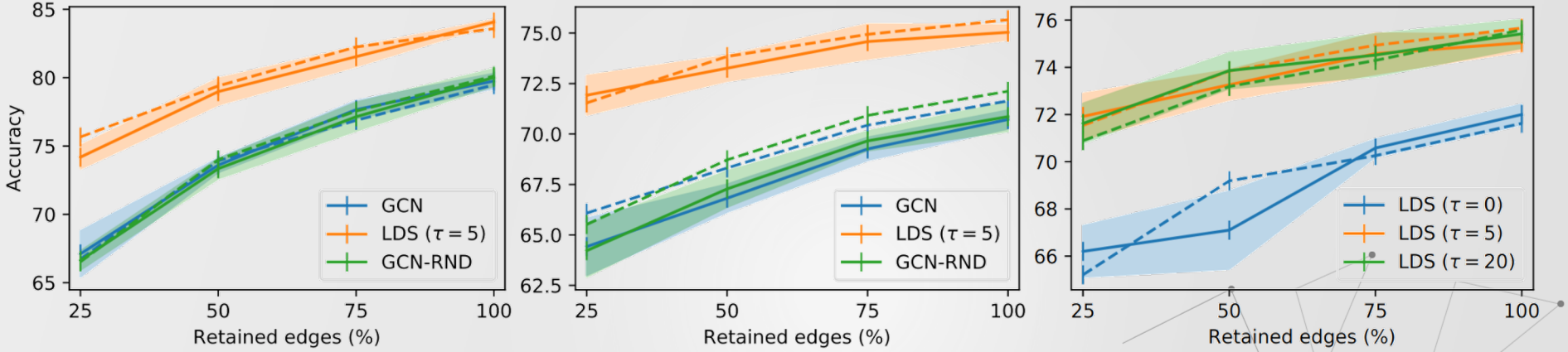


Figure 2. Mean accuracy $\pm$ standard deviation on validation (early stopping; dashed lines) and test (solid lines) sets for edge deletion scenarios on Cora (left) and Citeseer (center). (Right) Validation of the number of steps τ used to compute the STE hypergradient (Citeseer); $\tau = 0$ corresponds to alternating minimization. All results are obtained from five runs with different random seeds.

EXPERIMENT2: Baseline

1. Sparse-GCN: a sparse random graph
2. Dense-GCN: a dense graph with equal edge probabilities
3. RBF-GCN: a dense RBF kernel on the input features
4. kNN-GCN: a sparse k-nearest neighbor graph on the input features
5. FFNN: feed-forward neural networks

EXPERIMENT2

Table 1. Test accuracy ($\pm$ standard deviation) in percentage on various classification datasets. The best results and the statistical competitive ones (by paired t-test with $\alpha = 0.05$) are in bold. All experiments have been repeated with 5 different random seeds. We compare kNN-LDS to several supervised baselines and semi-supervised learning methods. No graph is provided as input. kNN-LDS achieves high accuracy results on most of the datasets and yields the highest gains on datasets with underlying graphs (Citeseer, Cora).

	Wine	Cancer	Digits	Citeseer	Cora	20news	FMA
LogReg	92.1 (1.3)	**93.3 (0.5)**	85.5 (1.5)	62.2 (0.0)	60.8 (0.0)	42.7 (1.7)	37.3 (0.7)
Linear SVM	93.9 (1.6)	**90.6 (4.5)**	87.1 (1.8)	58.3 (0.0)	58.9 (0.0)	40.3 (1.4)	35.7 (1.5)
RBF SVM	**94.1 (2.9)**	**91.7 (3.1)**	86.9 (3.2)	60.2 (0.0)	59.7 (0.0)	41.0 (1.1)	**38.3 (1.0)**
RF	93.7 (1.6)	**92.1 (1.7)**	83.1 (2.6)	60.7 (0.7)	58.7 (0.4)	40.0 (1.1)	**37.9 (0.6)**
FFNN	89.7 (1.9)	**92.9 (1.2)**	36.3 (10.3)	56.7 (1.7)	56.1 (1.6)	38.6 (1.4)	33.2 (1.3)
LP	89.8 (3.7)	76.6 (0.5)	**91.9 (3.1)**	23.2 (6.7)	37.8 (0.2)	35.3 (0.9)	14.1 (2.1)
ManiReg	90.5 (0.1)	81.8 (0.1)	83.9 (0.1)	67.7 (1.6)	62.3 (0.9)	**46.6 (1.5)**	34.2 (1.1)
SemiEmb	91.9 (0.1)	89.7 (0.1)	**90.9 (0.1)**	68.1 (0.1)	63.1 (0.1)	**46.9 (0.1)**	34.1 (1.9)
Sparse-GCN	63.5 (6.6)	72.5 (2.9)	13.4 (1.5)	33.1 (0.9)	30.6 (2.1)	24.7 (1.2)	23.4 (1.4)
Dense-GCN	90.6 (2.8)	90.5 (2.7)	35.6 (21.8)	58.4 (1.1)	59.1 (0.6)	40.1 (1.5)	34.5 (0.9)
RBF-GCN	90.6 (2.3)	**92.6 (2.2)**	70.8 (5.5)	58.1 (1.2)	57.1 (1.9)	39.3 (1.4)	33.7 (1.4)
kNN-GCN	93.2 (3.1)	**93.8 (1.4)**	**91.3 (0.5)**	68.3 (1.3)	66.5 (0.4)	41.3 (0.6)	**37.8 (0.9)**
kNN-LDS	**97.3 (0.4)**	**94.4 (1.9)**	**92.5 (0.7)**	**71.5 (1.1)**	**71.5 (0.8)**	**46.4 (1.6)**	**39.7 (1.4)**

EXPERIMENT3

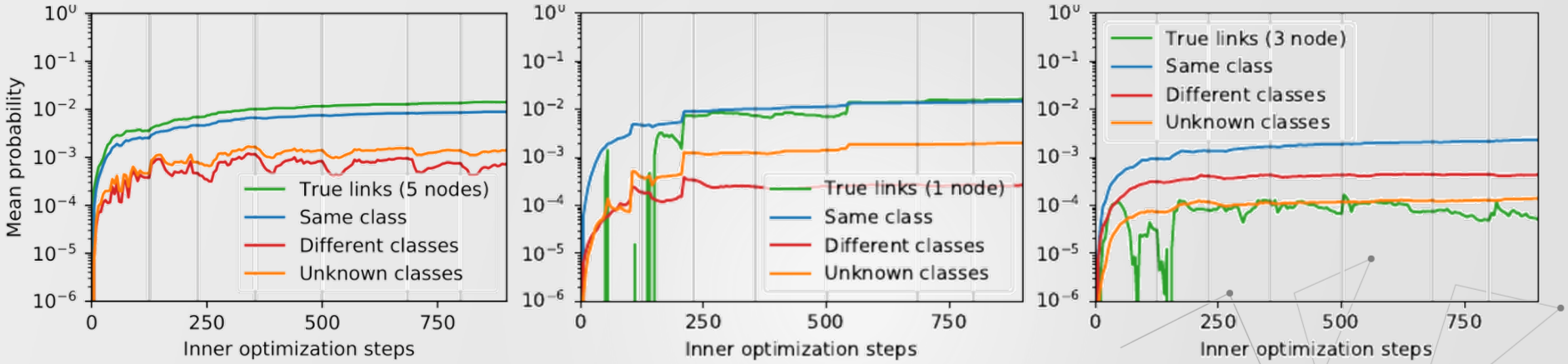


Figure 3. Mean edge probabilities to nodes aggregated w.r.t. four groups during LDS optimization, in log_{10} scale for three example nodes. For each example node, all other nodes are grouped by the following criteria: (a) adjacent in the ground truth graph; (b) same class membership; (c) different class membership; and (d) unknown class membership. Probabilities are computed with LDS ($\tau = 5$) on Cora with 25% retained edges. From left to right, the example nodes belong to the training, validation, and test set, respectively. The vertical gray lines indicate when the inner optimization dynamics restarts, that is, when the weights of the GCN are reinitialized.

LDS can learn highly non-uniform edge probabilities that reflect the class membership of the nodes.

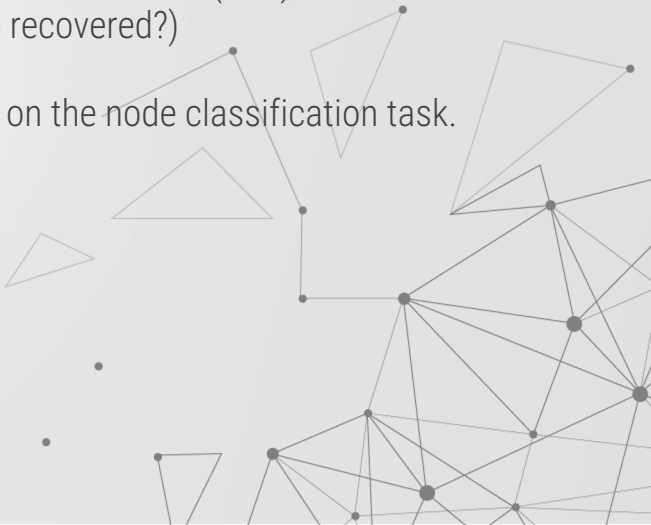
INSIGHTS

1. Classification problem can also use graph structure (learnable or kNN) to achieve more efficient and stable performance. (In practice, we find kNN graph is helpful for small dataset.)

2. To learn graph structure, an initial graph is needed since the search space of a graph is too large (2^{n*n}).

3. The authors only demonstrate the performance improvement of their method(LDS), do not show the ability of graph completion. (How many true links are recovered?)

4. Although authors have a general statement, they only test LDS on the node classification task. (For graph classification task, is LDS still useful?)



DISCRETE GRAPH STRUCTURE LEARNING FOR FORECASTING MULTIPLE TIME SERIES

Chao Shang, Jie Chen, and Jinbo Bi. ICLR 2021

Main Method Differences

LDS:

$$\min_{\theta \in \overline{\mathcal{H}}_N} \mathbb{E}_{A \sim \mathrm{Ber}(\theta)} \left[F(w_\theta, A) \right]$$
$$\text{such that } \; w_\theta = \arg\min_w \mathbb{E}_{A \sim \mathrm{Ber}(\theta)} \left[L(w, A) \right].$$

GTS:

$$\min_w \quad \mathrm{E}_{A \sim \mathrm{Ber}(\theta(w))} [F(A, w, X_{\mathrm{train}})].$$

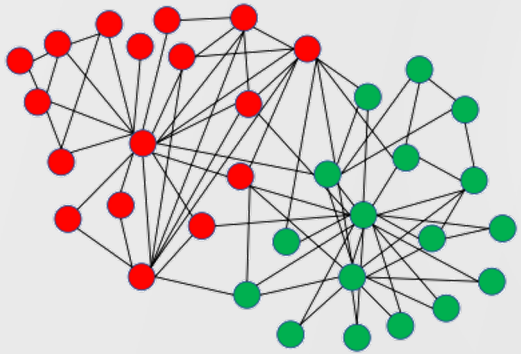
End2End Learning

Data Augmentation for Graph Neural Networks

Tong Zhao, Yozen Liu, Leonardo Neves, Oliver Woodford, Meng Jiang, Neil Shah. AAAI 2021

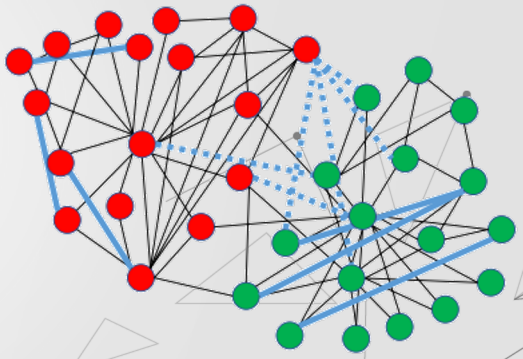
INTUITION

In the graph node classification task, there are two kinds of edges, intra-class edges and inter-class edges. Due to GNN message passing mechanism, adding intra-class edges and removing inter-class edges can make the node embedding from same classes more similar and improve node classification performance.



(a) Original graph.

What we want



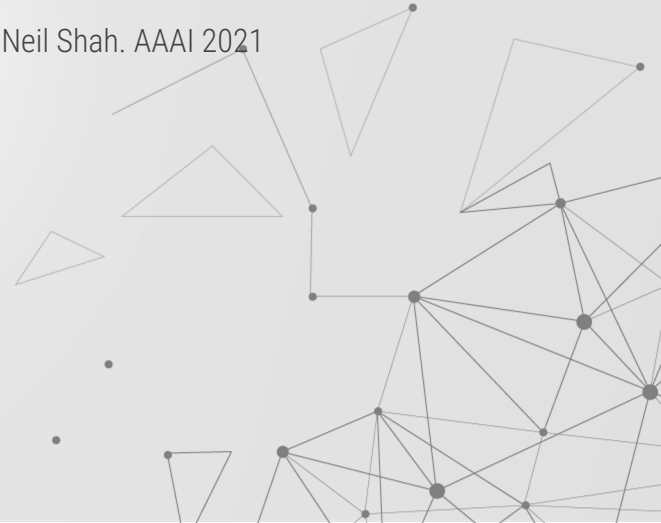
(d) Omniscient mod.

REFERENCE

1. Learning Discrete Structures for Graph Neural Networks
   Luca Franceschi, Mathias Niepert, Massimiliano Pontil, Xiao He. ICML 2019

2. Discrete Graph Structure Learning for Forecasting Multiple Time Series
   Chao Shang, Jie Chen, and Jinbo Bi. ICLR, 2021.

3. Data Augmentation for Graph Neural Networks
   Tong Zhao, Yozen Liu, Leonardo Neves, Oliver Woodford, Meng Jiang, Neil Shah. AAAI 2021



THANKS

Does anyone have any questions?