Learning Discrete Structures for Graph Neural Networks

Luca Franceschi, Mathias Niepert, Massimiliano Pontil, Xiao He

Presenter: Arshdeep Sekhon
https://qdata.github.io/deep2Read

Key Idea

Learning a **Discrete Graph** Structure along with **GCN** parameters for semi supervised node classification

not learn a graph that is similar to training graphs, rather learn a graph as a means to perform well on classification problem when input is not a collection of graphs

Ways to incorporate Relational Structure

- k-Nearest Neighbor between the datapoints
- kernel matrix between data points

Ways to incorporate Relational Structure

- k-Nearest Neighbor between the datapoints
    - (-) graph learning and end task disjoint
    - (-) lots of choices: k, similarity metric
- kernel matrix between data points
    - (-) dense dependency structure

Notations

- N nodes
- M edges
- Binary Adjacency matrix A
- Denote by $\mathcal{H}_N$ the space of all adjacency matrices (2^{N^2})
- $L = D - A$
- inputs: $X \in \mathcal{X}_N \subset R^{N \times n}$ where n is the number of node features
- A labeling function $y : V \to \mathcal{Y}$ where $\mathcal{Y}$ is the set of labels

GCNs

$$f_w : \mathcal{X}_N \times \mathcal{H}_N \rightarrow \mathcal{Y}^N \tag{1}$$

$$L(w, A) = \sum_{v \in V_{Train}} \ell(f_w(X, A)_v, y_v) + \Omega(w) \tag{2}$$

$w \in R^d$ are the parameters of f_w, Ω is a regularizer, $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathcal{R}^+$(?) is a loss function.

Bilevel Programs

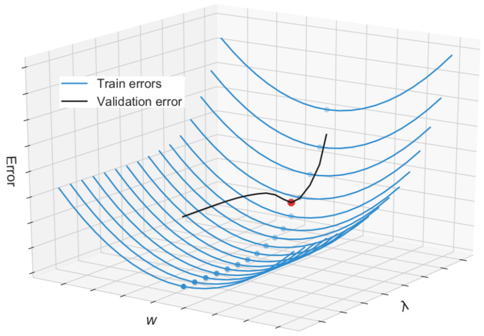
- Bilevel programs are mathematical programs with optimization problems in their constraints.
- hierarchical optimization problems where the feasible region of the so-called upper level problem is restricted by the graph of the solution set mapping of the lower level problem

$$\min_{\theta, w_\theta} F(\theta, w_\theta) \quad s.t. \quad w_\theta \in \arg\min_{w} L(w, \theta) \tag{3}$$

Motivation: Hyperparameter Optimization

Two levels:

- Nesting two search problems:
  - hyperparameter optimization: find a good hypothesis space
  - Supervised training: learn a good hypothesis in a hypothesis space



Bilevel Programming Objective

$$\min_{\theta, w_\theta} F(\theta, w_\theta) s.t. w_\theta \in \arg\min_{w} L(w, \theta) \tag{4}$$

Issue: closed form solution to inner objective not available

Formulation: Jointly Learning Structure and Parameters

- Find $A \in \mathcal{H}_N$ to minimize generalization error on a validation set V_{val}
    - $F(w_A, A) = \sum_{v \in V_{Val}} \ell(f_{w_A}(X, A)_v, y_v)$
- Here, w_A is the minimizer or (argmin) for $L(w, A)$ for a fixed adjacency matrix A.

$$L(w, A) = \sum_{v \in V_{Train}} \ell(f_w(X, A)_v, y_v) + \Omega(w) \tag{5}$$

Formulation: Jointly Learning Structure and Parameters

- Find $A \in \mathcal{H}_N$ to minimize generalization error on a validation set V_{val}
  - $F(w_A, A) = \sum_{v \in V_{Val}} \ell(f_{w_A}(X, A)_v, y_v)$
- Here, w_A is the minimizer or (argmin) for $L(w, A)$ for a fixed adjacency matrix A.

$$L(w, A) = \sum_{v \in V_{Train}} \ell(f_w(X, A)_v, y_v) + \Omega(w) \tag{5}$$

Final Objective Mixed Integer Bilevel Programming Problem

$$\min_{w_A, A} F(w_A, A) = \sum_{v \in V_{Val}} \ell(f_{w_A}(X, A)_v, y_v)$$

$$W_A \in \arg\min L(w, A) = \sum_{v \in V_{Train}} \ell(f_w(X, A)_v, y_v) + \Omega(w)$$

but

- Intractable to solve for even small graphs(2^{N^2})
- Optimizing A: continuous + discrete :

but

- Intractable to solve for even small graphs(2^{N^2})
- Optimizing A: continuous + discrete :
  - Continuous relaxation
  - **maintain a generative model for the graph structure : change A to parameters of the generative distribution**

Model A as a graph generative distribution

- $A \sim Ber(\theta)$ where $\theta \in \bar{\mathcal{H}_N}$, $\bar{\mathcal{H}_N} = ConvexHull(\mathcal{H}_N)$
- By modeling all possible edges as a set of mutually independent Bernoulli random variables with parameter matrix $\theta \in \overline{\mathcal{H}}_N$ we can sample graphs as $\mathcal{H}_N \ni A \sim \mathrm{Ber}(\theta)$.
- The resulting bilevel problem can be written as

$$\min_{\theta \in \overline{\mathcal{H}}_N} \mathbb{E}_{A \sim \mathrm{Ber}(\theta)} \left[F(w_\theta, A) \right] \tag{6}$$

$$\text{such that } \; w_\theta = \arg\min_w \mathbb{E}_{A \sim \mathrm{Ber}(\theta)} \left[L(w, A) \right]. \tag{7}$$

- both inner and outer now continuous functions of θ
- Still computationally expensive: No inner closed form solution(non convex), intractable exact expectations (2^{N^2})

GCN during testing

$$f_w^{exp}(X) = E_A([f_w(X, A)]) = \sum_{A \in \mathcal{H}_N} P_\theta(A) f_w(X, A) \tag{8}$$

Intractable (2^{N^2}), so compute empirical estimate:

$$\hat{f}_w(X) = \frac{1}{S} \sum_{i=1}^{S} f_w(X, A_i) \tag{9}$$

Advantages of formulating as a graph generative model

- sparse
- can be interpreted proabilistically

Optimizing the objective: Hypergradient Descent
A. Inner Objective

- Hypergradient: Used for hyperparameter optimization, treat graph generation θ as a hyperparameter
- the expectation

$$\mathbb{E}_{A\sim \mathrm{Ber}(\theta)}\left[L(w,A)\right] = \sum_{A\in\mathcal{H}_N} P_\theta(A)L(w,A) \tag{10}$$

  is composed of a sum of 2^{N^2} terms, which is intractable even for relatively small graphs.
- choose a tractable approximate learning dynamics Φ such as stochastic gradient descent (SGD),

$$w_{\theta,t+1} = \Phi(w_{\theta,t}, A_t) = w_{\theta,t} - \gamma_t \nabla L(w_{\theta,t}, A_t), \tag{11}$$

  where γ_t is a learning rate and $A_t \sim \mathrm{Ber}(\theta)$ is drawn at each iteration.
- Let $w_{\theta,T}$ be an approximate minimizer of $\mathbb{E}\left[L\right]$ (where T may depend on θ).

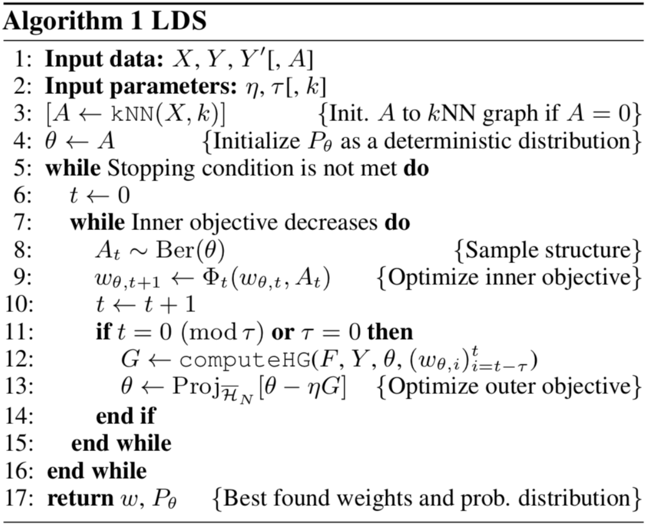
Optimizing the objective: Hypergradient Descent
A. Outer Objective

an estimator for the hypergradient $\nabla_\theta \mathbb{E}_{A\sim\mathrm{Ber}(\theta)}\left[F\right]$.

$$\begin{aligned}\nabla_\theta \mathbb{E}\left[F(w_{\theta,T},A)\right] = \mathbb{E}\left[\nabla_\theta F(w_{\theta,T},A)\right] = \\ \mathbb{E}\left[\partial_w F(w_{\theta,T},A)\nabla_\theta w_{\theta,T} + \partial_A F(w_{\theta,T},A)\nabla_\theta A\right],\end{aligned} \tag{12}$$

where we can swap the gradient and expectation operators since the expectation is over a finite random variable, assuming that the loss function F bounded. The second step is by the chain rule.

Optimization



Algorithm 1 LDS

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

Model Summary

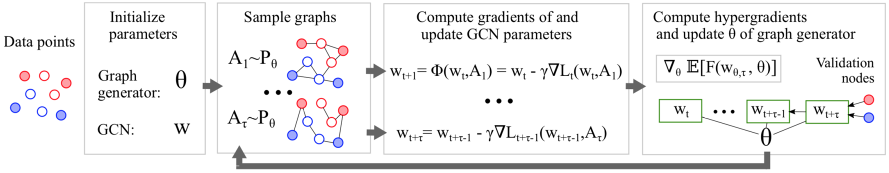

Data points
Initialize parameters
Graph generator: θ
GCN: w
Sample graphs
$A_1 \sim P_\theta$
$A_\tau \sim P_\theta$
Compute gradients of and update GCN parameters
$w_{t+1} = \Phi(w_t, A_1) = w_t - \gamma \nabla L_t(w_t, A_1)$
$w_{t+\tau} = w_{t+\tau-1} - \gamma \nabla L_{t+\tau-1}(w_{t+\tau-1}, A_\tau)$
Compute hypergradients and update θ of graph generator
$\nabla_\theta \; \mathbb{E}[F(w_{\theta,\tau}, \theta)]$
Validation nodes
w_t
$w_{t+\tau-1}$
$w_{t+\tau}$
θ

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

Experiments 1: Graphs with Missing Edges

- All GCN 16 hidden units + ReLU
- For LDS, split validation set for validation vs outer stopping sets
- available tensor flow implementation

Experiments 1: Graphs with Missing Edges

Baselines

- GCN
- GCN-RND: add randomly sampled edges at each optimization of a vanilla gcn[1]

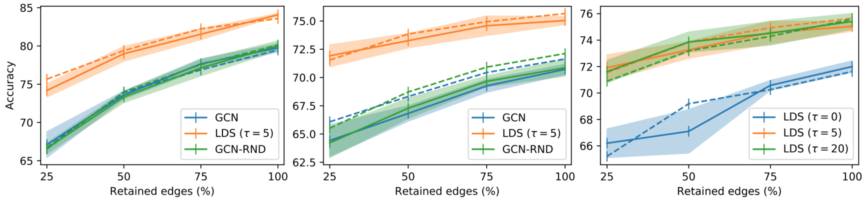
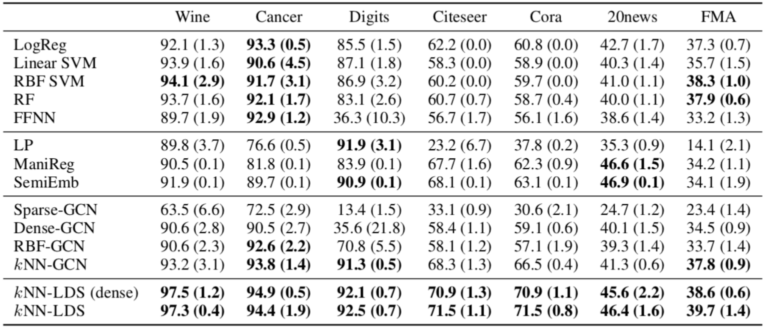


Figure 2. Mean accuracy ± standard deviation on validation (early stopping; dashed lines) and test (solid lines) sets for edge deletion scenarios on Cora (left) and Citeseer (center). (Right) Validation of the number of steps τ used to compute the hypergradient (Citeseer); $\tau = 0$ corresponds to alternating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

[1]to show adding random edges does not improve generalization

Experiments 2: Semi Supervised Classification



	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 (dense)	**97.5 (1.2)**	**94.9 (0.5)**	**92.1 (0.7)**	**70.9 (1.3)**	**70.9 (1.1)**	**45.6 (2.2)**	**38.6 (0.6)**
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)**

Experiments 3: optimization Dynamics

- capture a useful distribution rather than pick exact links

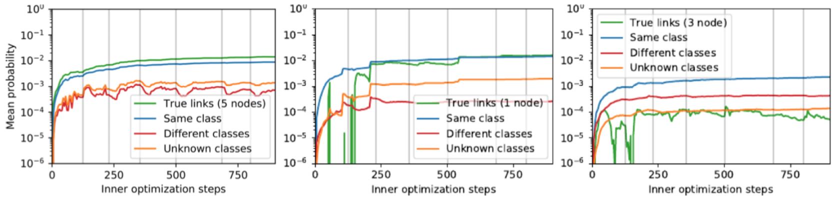


Figure 3. Mean edge probabilities to nodes aggregated w.r.t. four groups during LDS optimization, in log_{10} scale for three example nodes. For each example node, all other nodes are grouped by the following criteria: (a) adjacent in the ground truth graph; (b) same class membership; (c) different class 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.

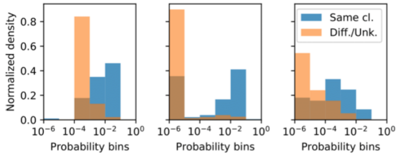


Figure 4. Normalized histograms of edges' probabilities for the same nodes of Figure 3.

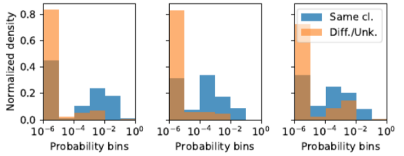


Figure 5. Histograms for three Citeseer test nodes, missclassified by kNN-GCN and rightly classified by kNN-LDS.

Related Work: Graph Synthesis and Generation

- ER model/ or other graph generative models: not learn a graph that is similar tot raining graphs, rather learn a graph as a means to perform well on classification problem when input is not a collection of graphs
- NRI: limited to dynamic interaction systems, unsupervised(no y)
- No true graph available

Related Work: Link Prediction

- Type 1: only link prediction from node similarity
- Type 2: Statistical Relational Learning: intractable, structure and parameter learning steps are separate

Conclusion: Drawbacks

- not scalable
- only transductive setting
- no extra prior on graphs: connected
- ?? True edges not evaluated

Extra Algorithm

Algorithm 2 LDS (extended)

```
Input data: $X, Y, Y'[, A]$
Input parameters: $\eta, \tau[, k]$
$[A \leftarrow \mathrm{kNN}(X, k)]$        {Init. $A$ to $k$NN graph if $A = 0$}
$\theta \leftarrow A$        {Initialize $P_\theta$ as a deterministic distribution}
while Stopping condition is not met do
    $t \leftarrow 0$
    while Inner objective decreases do
        $A_t \sim \mathrm{Ber}(\theta)$        {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
            $A_t \sim \mathrm{Ber}(\theta)$
            $p \leftarrow \partial_w F(w_{\theta,t}, A_t)$
            $G \leftarrow \partial_A F(w_{\theta,t}, A_t)$
            for $s = t - 1$ downto $t - \tau$ do
                $A_s \sim \mathrm{Ber}(\theta)$
                $p \leftarrow p D_s(w_{\theta,s}, A_s)$
                $G \leftarrow G + p E_s(w_{\theta,s}, A_s)$
            end for
            [illegible]
```