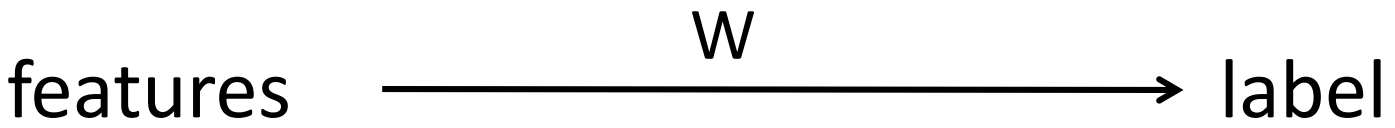
Learning Discrete Structures for Graph Neural Networks

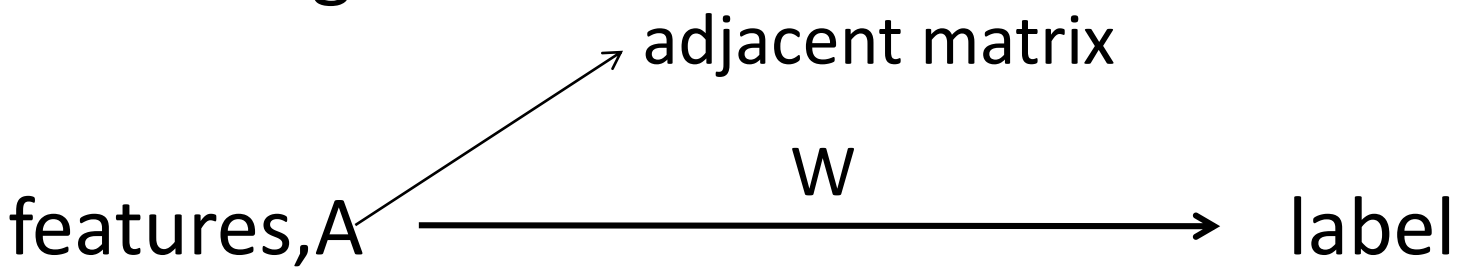
Luca Franceschi Mathias Niepert Massimiliano Pontil Xiao He

ICML 2019

DNN Learning:



GNN Learning:



A is missing or incomplete

KNN:efficacy of the resulting models hinges on the choice of k

Bilevel programs:

a set of variables occurring in the objective function are constrained to be an optimal solution of another optimization problem

$$\theta \in \mathbb{R}^m \qquad\qquad w \in \mathbb{R}^d$$

$$\min_{\theta, w_\theta} F(w_\theta, \theta) \text{ such that } w_\theta \in \arg\min_{w} L(w, \theta).$$

replacing the minimization of L with the repeated application of an iterative optimization dynamics Φ such as (stochastic) gradient descent

$$w_t = \Phi(w_{t-1}, \theta) = w_{t-1} - \eta \nabla L(w_{t-1}, \theta), t = 1, 2 \ldots\ldots T$$

hypergradient

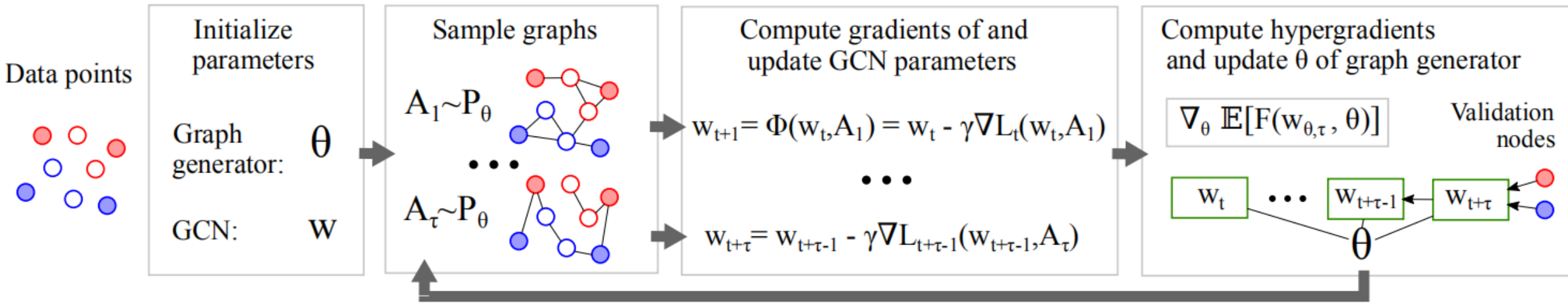
$$\nabla_\theta F(w_{\theta,T}, \theta) = \partial_w F(w_{\theta,T}, \theta) \nabla_\theta w_{\theta,T} + \partial_\theta F(w_{\theta,T}, \theta)$$

$$\theta = \theta - \eta_{outer} \nabla_\theta F(w_{\theta,T}, \theta)$$

The original loss function of GCN is

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

$$F(w_A, A) = \sum_{v \in V_{\texttt{Val}}} \ell(f_{w_A}(X, A)_v, y_v), \longrightarrow \theta \in \mathbb{R}^m$$


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

$$\partial_w F(w_{\theta,T}, \theta) \nabla_\theta w_{\theta,T} + \partial_\theta F(w_{\theta,T}, \theta),$$

Moreover, it contains both continuous and discrete-valued variables, which prevents us from directly applying

we propose to model each edge with a Bernoulli random variable

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

$$\frac{\mathrm{d}w_t}{\mathrm{d}\theta} = \frac{\partial \Phi(w_{t-1}, \theta)}{\partial w_{t-1}} \frac{\mathrm{d}w_{t-1}}{\mathrm{d}\theta} + \frac{\partial \Phi(w_{t-1}, \theta)}{\partial \theta}$$

- $A_t = \frac{\partial \Phi_t(w_{t-1}, \theta)}{\partial w_{t-1}}$ (6)
- $B_t = \frac{\partial \Phi_t(w_{t-1}, \theta)}{\partial \theta}$ (7)

$$Z_t = A_t Z_{t-1} + B_t$$

最后计算出：

- $\nabla_\theta F(w_\theta, \theta) = \nabla F(w_T) \sum_{t=1}^{T} (\prod_{s=t+1}^{T} A_s) B_t$ (8)

Algorithm 1 LDS

Input data: X, Y, Y'[, A]
Input parameters: η, τ[, 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}

Expriments

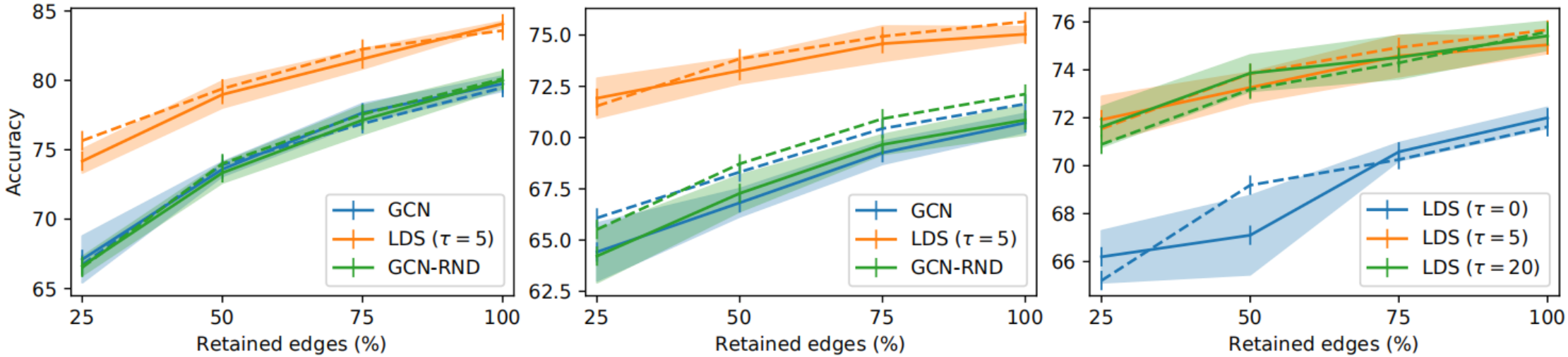


Figure 2. Mean accuracy $\pm$ standard deviation on validation (early stopping; dashed lines) and test (solid lines) sets for edge deletion scenarios on Cora (left) and Citeseer (center). (Right) Validation of the number of steps τ used to compute the hypergradient (Citeseer); $\tau = 0$ corresponds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

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 (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)**

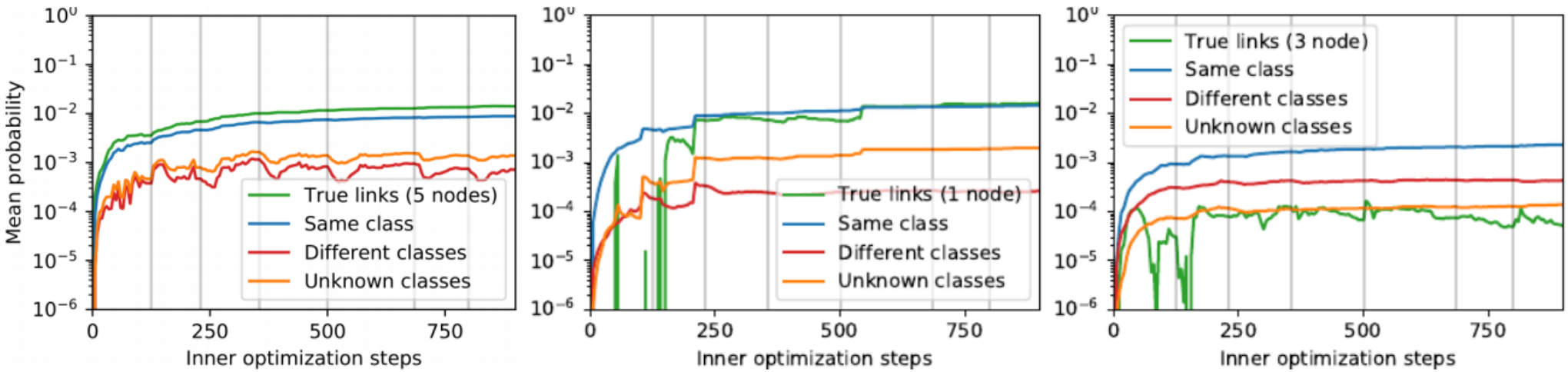


Figure 3. Mean edge probabilities to nodes aggregated w.r.t. four groups during LDS optimization, in log_{10} scale for three example nodes. For each example node, all other nodes are grouped by the following criteria: (a) adjacent in the ground truth graph; (b) same class membership; (c) different class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