Graph Structure Learning for GCNs

Luca Franceschi[1,3], Mathias Niepert[2], Massimiliano Pontil[1,3], Xiao He[2]

1. Istituto Italiano di Tecnologia, Genoa, IT | 2. NEC Labs Europe, Heidelberg, DE | 3. University College London, UK

Overview

We propose a method, named **LDS**, for jointly learning the graph and weights of a GCN by approximately solving a bilevel program that **learns a discrete probability distribution** on the edges of the graph.
LDS makes it possible to apply GCNs to real-world scenarios where the given graph is incomplete or corrupted, and even when the graph is not available at all.
LDS outperforms related approaches by a significant margin on datasets where the dependency structure is either incomplete or completely missing.

Jointly learning the structures and the parameters

We are interested in learning models of the type

$$f_w : \mathcal{X}_N \times \mathcal{H}_N \to \mathcal{Y}^N; \quad f_w(X, A) = \hat{y}$$

where $w \in \mathbb{R}^d$ are the parameters, $\mathcal{X}_N$ and $\mathcal{Y}^N$ are input and output spaces, and $\mathcal{H}_N = \{0,1\}^{N \cdot N}$ is the space of undirected adjacency matrices.
Parameters are learned by minimizing

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

If A is unknown, one may learn it by minimizing

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

The resulting bilevel problem is hard to solve.

We reformulate it by introducing a *discrete* generative model for the graph as independent Bernoullis with parameter $\theta \in [0,1]^{N \cdot N}$:

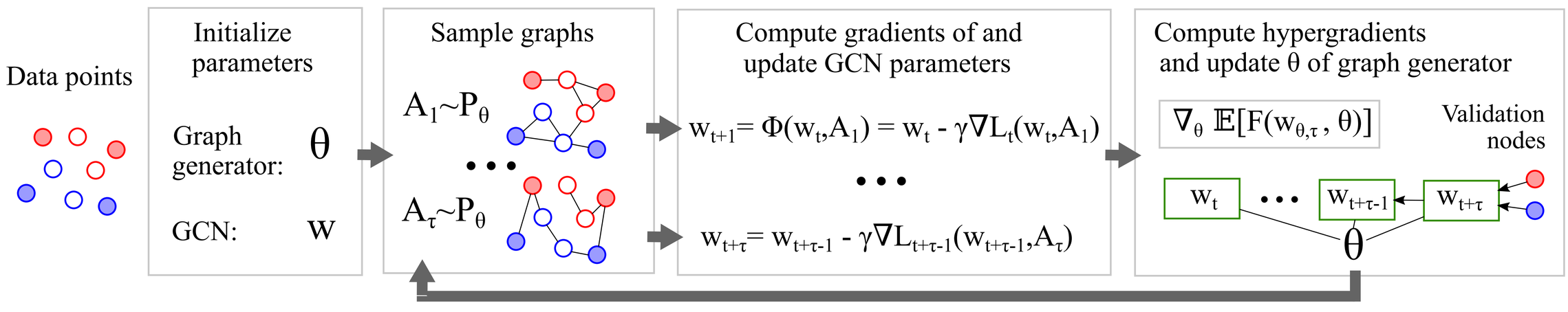
$$\min_{\theta \in \overline{\mathcal{H}}_N} \mathbb{E}_{A \sim \mathrm{Ber}(\theta)} [F(w_\theta, A)] \quad (1)$$

$$\text{s. t.} \quad w_\theta = \arg\min_w \mathbb{E}_{A \sim \mathrm{Ber}(\theta)} [L(w, A)] \quad (2)$$

Approximating the inner problem with repeated application of a stochastic optimization dynamics Φ, the (hyper)gradient $\nabla_\theta \mathbb{E}[F(w_{\theta,T}, A)]$ is

$$\mathbb{E}[\partial_w F(w_{\theta,T}, A) \nabla_\theta w_{\theta,T} + \partial_A F(w_{\theta,T}, A) \nabla_\theta A]$$

We use the straight-through estimator and (truncated) reverse-mode differentiation.



Introduction & Motivation

- Relational learning is effective when the predictive model can leverage relationships among data points.
- Relations, however, should be given to the learning algorithm e.g. in the form of graph.
- Often, in practice, graphs may be unavailable, or only partially known.
- **Common heuristic:** build a kNN graph from the data. A **shortcoming** it is a trial-and-error procedure that depends on the choice of hyperparameters (e.g. metrics, k, ...).
- **Contribution:** we learn a *discrete* and *sparse* dependency structure between data points and simultaneously optimize the parameters of a **GCN** (graph convolutional neural network) by approximately solving a **bilevel programming problem**.

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**
    $A_t \sim \mathrm{Ber}(\theta)$ {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_θ {Best found weights and prob. distribution}

Experiments: Missing Edges

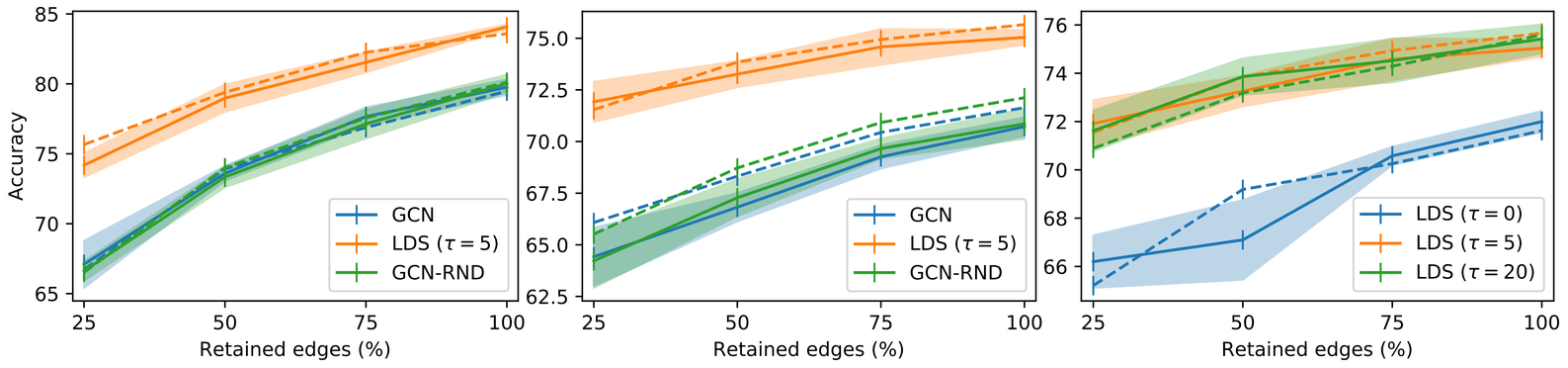


Figure 1: Mean accuracy ± standard deviation on validation and test sets (dashed and solid lines) for missing edges scenario on Cora (left) and Citeseer (center). Right: Validation of the number of steps τ used to compute the hypergradient (Citeseer).

Experiments: Semi-supervised Learning (no graph)

Table 1: Test accuracy (± standard deviation, 5 random seeds) in percentage on various classification datasets. We compare kNN-LDS to several supervised baselines and semi-supervised learning methods. No graph is provided as input.

	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)
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)**
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	88.2 (3.3)	86.0 (4.0)	82.8 (3.3)	67.7 (1.6)	62.3 (0.9)	**46.6 (1.5)**	34.2 (1.1)
SemiEmb	88.9 (3.7)	86.4 (5.4)	**90.7 (2.7)**	68.1 (0.7)	62.9 (1.1)	**46.3 (1.6)**	35.5 (1.9)
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	93.1 (2.7)	**93.4 (2.8)**	**91.3 (2.0)**	69.5 (1.1)	66.5 (0.4)	43.3 (0.4)	**37.9 (0.5)**
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)**

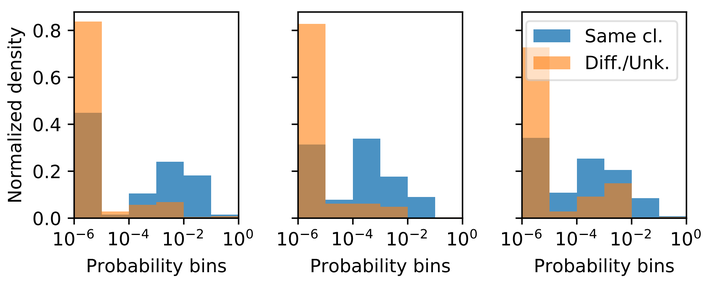


Figure 2: Histograms for three Citeseer test nodes, missclassified by kNN-GCN and rightly classified by kNN-LDS.

Conclusions & Future Work

- LDS makes GCNs **successfully applicable to new settings**, by learning sparse and discrete relationships among data points
- *Limitations*: no inductive setting, medium-small graphs (must scale up to larger datasets)
- Explore more complex generative models

- Franceschi, Luca, et al. *"Bilevel programming for hyperparameter optimization and meta-learning"*, ICML (2018)
- Kipf, Thomas N., Max Welling *"Semi-supervised classification with graph convolutional networks"*, ICLR (2017)
- Bengio, Yoshua, et al. *"Estimating or propagating gradients through stochastic neurons for conditional computation"*, (2013)
- **Full paper:** Franceschi, Luca et al. *"Learning Discrete Structures for Graph Neural Networks"* ICML (2019)