TEXAS A&M UNIVERSITY

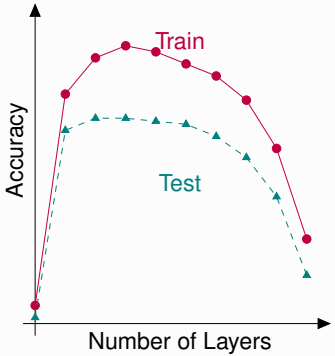
Bayesian Graph Neural Networks with Adaptive Connection Sampling

Arman Hasanzadeh, Ehsan Hajiramezanali, Shahin Boluki,
Mingyuan Zhou (UT-Austin), Nick Duffield, Krishna Narayanan, Xiaoning Qian

37-th International Conference on Machine Learning (ICML)

GCN limitations

- Over-smoothing
  - due to the nature of low-pass filtering of GCNs
  - training accuracy can drop with the number of layers, in even shallow networks
- No uncertainty quantification
  - due to deterministic modeling
  - lack of Bayesian interpretation
- Existing stochastic regularization and reduction techniques
  - DropOut, DropEdge, node sampling, random walk
  - not very successful in alleviating over-smoothing
  - no Bayesian approximation interpretation
  - difficulty in hand-tuning of hyper-parameters



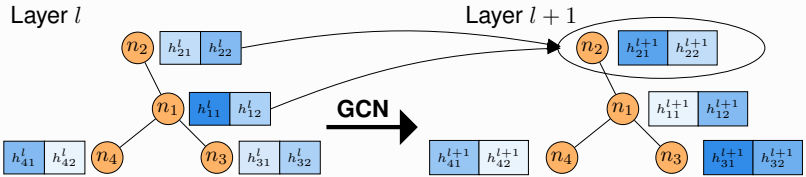
Graph DropConnect (GDC); overview

TEXAS A&M UNIVERSITY

We propose GDC, a unified framework for stochastic regularization and reduction:

- generalization of existing GCN regularization methods
- enabling natural Bayesian interpretation and prediction with uncertainty quantification
- learnable drop rates
  - drop rates and masks are modeled as beta and Bernoulli random variables
  - inferring drop rates with variational inference, coupled with GCN training:
    - approximate optimization with concrete relaxation
    - direct optimization using Augment-REINFORCE-Merge (ARM)
- better regularization for GCNs by alleviating over-smoothing

Review; GCN

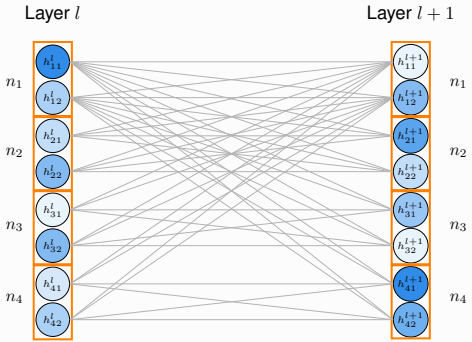


$$\mathbf{H}^{(l+1)} = \sigma\left(\mathfrak{N}(\mathbf{A})\,\mathbf{H}^{(l)}\,\mathbf{W}^{(l)}\right)$$

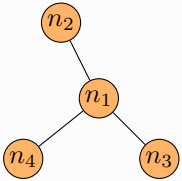
- Connections are localized

$\mathbf{A}$ = adjacency matrix
$\mathfrak{N}$ = normalizing operator
$\mathbf{W}$ = weight matrix
$\mathbf{H}^{(l)}$ = hidden layer l
σ = activation function

Review; GCN, contd.

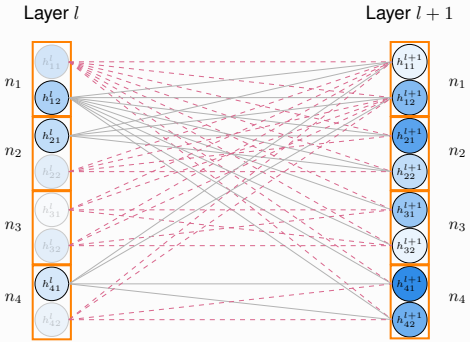


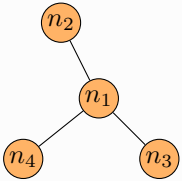
$$\mathbf{H}^{(l+1)} = \sigma\left(\mathfrak{N}(\mathbf{A})\,\mathbf{H}^{(l)}\,\mathbf{W}^{(l)}\right)$$



$\mathbf{A}$ = adjacency matrix
$\mathfrak{N}$ = normalizing operator
$\mathbf{W}$ = weight matrix
$\mathbf{H}^{(l)}$ = hidden layer l
σ = activation function

Review; DropOut

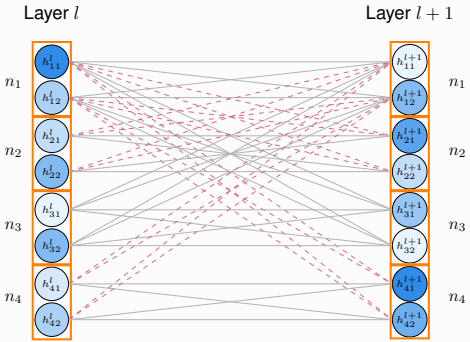




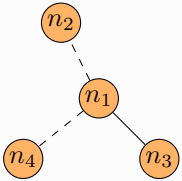
$$\mathbf{H}^{(l+1)} = \sigma\left(\mathfrak{N}(\mathbf{A})(\mathbf{Z}^{(l)} \odot \mathbf{H}^{(l)})\,\mathbf{W}^{(l)}\right)$$

$\mathbf{A}$ = adjacency matrix
$\mathfrak{N}$ = normalizing operator
$\mathbf{W}$ = weight matrix
$\mathbf{H}^{(l)}$ = hidden layer l
$\mathbf{Z}^{(l)}$ = Bernoulli mask, layer l
σ = activation function

Review; DropEdge

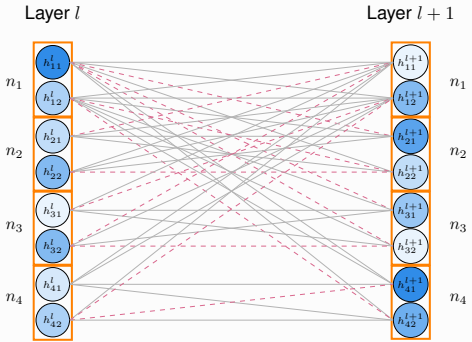


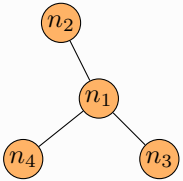
$$\mathbf{H}^{(l+1)} = \sigma\left(\mathfrak{N}(\mathbf{A} \odot \mathbf{Z}^{(l)})\, \mathbf{H}^{(l)}\, \mathbf{W}^{(l)}\right)$$



$\mathbf{A}$ = adjacency matrix
$\mathfrak{N}$ = normalizing operator
$\mathbf{W}$ = weight matrix
$\mathbf{H}^{(l)}$ = hidden layer l
$\mathbf{Z}^{(l)}$ = Bernoulli mask, layer l
σ = activation function

Proposed method; GDC





$$\mathbf{H}^{(l+1)}[:,j] = \sigma\left(\sum_{i=1}^{f_l} \mathfrak{N}(\mathbf{A} \odot \mathbf{Z}^{(l)}_{i,j})\, \mathbf{H}^{(l)}[:,i]\, \mathbf{W}^{(l)}[i,j]\right)$$

$\mathbf{A}$ = adjacency matrix
$\mathfrak{N}$ = normalizing operator
$\mathbf{W}$ = weight matrix
$\mathbf{H}^{(l)}$ = hidden layer l
$\mathbf{Z}^{(l)}_{i,j}$ = Bernoulli mask, layer l
f_l = # of features, layer l
σ = activation function

GDC as a Bayesian approximation

TEXAS A&M UNIVERSITY

- Bayesian neural network (BNN):
  - Weights are random variables
- Graph DropConnect: randomness in adjacency matrix
  - How to transform randomness from adjacency to weights?
  - How to choose prior and posterior such that BNN loss is equal to GDC loss?

$$\mathcal{L}_{\text{GDC}} = \frac{1}{|\mathcal{O}|} \sum_{i \in \mathcal{O}} E(y_i,\, \hat{y}_i) + \lambda \sum_{l=1}^{L} ||\mathbf{W}^{(l)}||_2^2$$

$\mathcal{O}$ = set of nodes with observed labels

GDC as a Bayesian approximation, contd.

TEXAS A&M UNIVERSITY

- For brevity, assume $\mathbf{Z}_{i,j}^{(l)}$ are the same for j's & rewrite GDC equation for each node
- Distribution over masks: $\mathbf{z}_{vu}^{(l)} \sim \mathrm{Ber}(\pi_l)$

$$\Rightarrow \mathbf{h}_v^{(l+1)} = \sigma\left(\frac{1}{c_v}\Big(\sum_{u\in\mathcal{N}(v)\cup\{v\}} \mathbf{h}_u^{(l)}\,\mathrm{diag}(\mathbf{z}_{vu}^{(l)})\Big)\,\mathbf{W}^{(l)}\right)$$

GDC as a Bayesian approximation, contd.

TEXAS A&M UNIVERSITY

- For brevity, assume $\mathbf{Z}_{i,j}^{(l)}$ are the same for j's & rewrite GDC equation for each node
- Distribution over masks: $\mathbf{z}_{vu}^{(l)} \sim \mathrm{Ber}(\pi_l)$

$$\Rightarrow \mathbf{h}_v^{(l+1)} = \sigma\left(\frac{1}{c_v}\Big(\sum_{u\in\mathcal{N}(v)\cup\{v\}} \mathbf{h}_u^{(l)}\,\mathrm{diag}(\mathbf{z}_{vu}^{(l)})\Big)\,\mathbf{W}^{(l)}\right)$$

$$= \sigma\left(\frac{1}{c_v}\sum_{u\in\mathcal{N}(v)\cup\{v\}} \mathbf{h}_u^{(l)}\,\big(\mathrm{diag}(\mathbf{z}_{vu}^{(l)})\,\mathbf{W}^{(l)}\big)\right) = \sigma\left(\frac{1}{c_v}\sum_{u\in\mathcal{N}(v)\cup\{v\}} \mathbf{h}_u^{(l)}\,\mathbf{W}_{vu}^{(l)}\right)$$

GDC as a Bayesian approximation, contd.

TEXAS A&M UNIVERSITY

- For brevity, assume $\mathbf{Z}_{i,j}^{(l)}$ are the same for j's & rewrite GDC equation for each node
- Distribution over masks: $\mathbf{z}_{vu}^{(l)} \sim \mathrm{Ber}(\pi_l)$

$$\Rightarrow \mathbf{h}_v^{(l+1)} = \sigma\left(\frac{1}{c_v}\Big(\sum_{u\in\mathcal{N}(v)\cup\{v\}} \mathbf{h}_u^{(l)}\,\mathrm{diag}(\mathbf{z}_{vu}^{(l)})\Big)\,\mathbf{W}^{(l)}\right)$$

$$= \sigma\left(\frac{1}{c_v}\sum_{u\in\mathcal{N}(v)\cup\{v\}} \mathbf{h}_u^{(l)}\,\left(\mathrm{diag}(\mathbf{z}_{vu}^{(l)})\,\mathbf{W}^{(l)}\right)\right) = \sigma\left(\frac{1}{c_v}\sum_{u\in\mathcal{N}(v)\cup\{v\}} \mathbf{h}_u^{(l)}\,\mathbf{W}_{vu}^{(l)}\right)$$

- Discrete prior distribution: $p(\mathbf{W}_{vu}^{(l)}) \propto \mathbf{e}^{\frac{1}{2}\mathbf{W}_{vu}^{(l)}\,\mathbf{W}_{vu}^{(l)^T}}$
- Posterior distribution: $q_{\theta_l}(\mathbf{W}_{vu}^{(l)}) = \pi_l\,\delta(\mathbf{W}_{vu}^{(l)} - \mathbf{0}) + (1-\pi_l)\,\delta(\mathbf{W}_{vu}^{(l)} - \mathbf{M}^{(l)})$

Variational inference for GDC

TEXAS A&M UNIVERSITY

- Loss function:

$$\mathbb{E}_{q(\{\boldsymbol{\omega}^{(l)},\, \mathbf{Z}^{(l)}\}_{l=1}^{L})}\left[\log P(Y_o \,|\, X, \{\boldsymbol{\omega}^{(l)},\, \mathbf{Z}^{(l)}\}_{l=1}^{L})\right] - \sum_{l=1}^{L} \mathrm{KL}\left(q(\boldsymbol{\omega}^{(l)},\, \mathbf{Z}^{(l)}) \,||\, p(\boldsymbol{\omega}^{(l)},\, \mathbf{Z}^{(l)})\right)$$

$Y_o =$ available labels for the observed nodes

Variational inference for GDC

TEXAS A&M UNIVERSITY

- Loss function:

$$\mathbb{E}_{q(\{\boldsymbol{\omega}^{(l)},\, \mathbf{Z}^{(l)}\}_{l=1}^{L})}\left[\log P(Y_o \,|\, X, \{\boldsymbol{\omega}^{(l)},\, \mathbf{Z}^{(l)}\}_{l=1}^{L})\right] - \sum_{l=1}^{L} \mathrm{KL}\left(q(\boldsymbol{\omega}^{(l)},\, \mathbf{Z}^{(l)}) \,||\, p(\boldsymbol{\omega}^{(l)},\, \mathbf{Z}^{(l)})\right)$$

$Y_o =$ available labels for the observed nodes

- KL divergence:

$$\mathrm{KL}\left(q(\boldsymbol{\omega}^{(l)}, \mathbf{Z}^{(l)}) \,||\, p(\boldsymbol{\omega}^{(l)}, \mathbf{Z}^{(l)})\right) \propto \frac{|\mathcal{E}|\,(1-\pi_l)}{2}\,||\mathbf{M}^{(l)}||_2^2 - \mathcal{H}(\pi_l) + \sum_{e=1}^{|\mathcal{E}|} \mathrm{KL}\left(q(\mathbf{z}_e^{(l)}) \,||\, p(\mathbf{z}_e^{(l)})\right)$$

$\mathcal{H}(\pi_l) =$ Bernoulli entropy $\qquad \mathcal{E} =$ set of edges + self-loops

Variational inference for GDC

TEXAS A&M UNIVERSITY

- Loss function:

$$\mathbb{E}_{q(\{\boldsymbol{\omega}^{(l)},\, \mathbf{Z}^{(l)}\}_{l=1}^{L})}\left[\log P(Y_o \,|\, X, \{\boldsymbol{\omega}^{(l)},\, \mathbf{Z}^{(l)}\}_{l=1}^{L})\right] - \sum_{l=1}^{L} \mathrm{KL}\left(q(\boldsymbol{\omega}^{(l)},\, \mathbf{Z}^{(l)}) \,||\, p(\boldsymbol{\omega}^{(l)},\, \mathbf{Z}^{(l)})\right)$$

$Y_o =$ available labels for the observed nodes

- KL divergence:

$$\mathrm{KL}\left(q(\boldsymbol{\omega}^{(l)}, \mathbf{Z}^{(l)}) \,||\, p(\boldsymbol{\omega}^{(l)}, \mathbf{Z}^{(l)})\right) \propto \frac{|\mathcal{E}|\,(1-\pi_l)}{2}\,||\mathbf{M}^{(l)}||_2^2 - \mathcal{H}(\pi_l) + \sum_{e=1}^{|\mathcal{E}|} \mathrm{KL}\left(q(\mathbf{z}_e^{(l)}) \,||\, p(\mathbf{z}_e^{(l)})\right)$$

$\mathcal{H}(\pi_l) =$ Bernoulli entropy $\qquad \mathcal{E} =$ set of edges + self-loops

- For fixed π_l's, a GCN with GDC & ℓ_2 norm regularizer approximates a Bayesian NN!

Corollary

Any graph neural network with random walk sampling, such as GraphSAGE, is an approximation of a Bayesian graph neural network as long as outputs are calculated using Monte-Carlo sampling.

Learnable GDC

TEXAS A&M UNIVERSITY

- High sparsity is required to achieve good performance [Rong et. al., 2019]
  - We impose a beta-Bernoulli prior to the binary random masks
- Assumption: drop masks are independent across features
  - Following equations are for $f_l = 1$
- Prior distribution:

$$z_e^{(l)} \sim \text{Bernoulli}(\pi_l), \qquad \pi_l \sim \text{Beta}(c/L,\, c(L-1)/L)$$

$c =$ constant

- Posterior distribution (Kumaraswamy distribution):

$$q(\mathbf{Z}^{(l)}, \pi_l) = \prod_{e=1}^{|\mathcal{E}|} q(z_e^{(l)} \mid \pi_l)\, q(\pi_l); \qquad q(\pi_l; a_l, b_l) = a_l b_l \pi_l^{a_l - 1}(1 - \pi_l^{a_l})^{b_l - 1}$$

$a_l > 0,\, b_l > 0 =$ variational parameters

Learnable GDC, contd.

TEXAS A&M UNIVERSITY

- Loss function:

$$\mathbb{E}_{q(\{\boldsymbol{\omega}^{(l)},\, \mathbf{Z}^{(l)}\}_{l=1}^{L})} \left[\log P(Y_o \,|\, X, \{\boldsymbol{\omega}^{(l)},\, \mathbf{Z}^{(l)}\}_{l=1}^{L}) \right] + \frac{|\mathcal{E}|\,(1-\pi_l)}{2} ||\mathbf{M}^{(l)}||_2^2 - \sum_{e=1}^{|\mathcal{E}|} \mathrm{KL}\left(q(\mathbf{z}_e^{(l)},\, \pi_l) \,||\, p(\mathbf{z}_e^{(l)},\, \pi_l) \right)$$

Learnable GDC, contd.

TEXAS A&M UNIVERSITY

- Loss function:

$$\mathbb{E}_{q(\{\boldsymbol{\omega}^{(l)},\, \mathbf{Z}^{(l)}\}_{l=1}^{L})}\left[\log P(Y_o \,|\, X, \{\boldsymbol{\omega}^{(l)},\, \mathbf{Z}^{(l)}\}_{l=1}^{L})\right] \quad + \quad \frac{|\mathcal{E}|\,(1-\pi_l)}{2}\,||\mathbf{M}^{(l)}||_2^2$$

$$- \quad \sum_{e=1}^{|\mathcal{E}|} \mathrm{KL}\left(q(\mathbf{z}_e^{(l)},\, \pi_l)\,||\,p(\mathbf{z}_e^{(l)},\, \pi_l)\right)$$

- KL divergence:

$$\mathrm{KL}\left(q(\mathbf{Z}^{(l)}, \pi_l)\,||\,p(\mathbf{Z}^{(l)}, \pi_l)\right) = \sum_{e=1}^{|\mathcal{E}|} \mathrm{KL}\left(q(z_e^{(l)}\,|\,\pi_l)\,||\,p(z_e^{(l)}\,|\,\pi_l)\right) + \mathrm{KL}\left(q(\pi_l)\,||\,p(\pi_l)\right),$$

$$\mathrm{KL}\left(q(\pi_l)\,||\,p(\pi_l)\right) = \frac{a_l - c/L}{a_l}\left(-\gamma - \Psi(b_l) - \frac{1}{b_l}\right) + \log\frac{a_l b_l}{c/L} - \frac{b_l - 1}{b_l}$$

$\gamma =$ Euler-Mascheroni constant, $\Psi(\cdot) =$ digamma function

Optimization of learnable GDC

TEXAS A&M UNIVERSITY

- Masks are discrete random variables $\Rightarrow$ reparametrization trick cannot be used
  - Solution 1: use concrete relaxation of Bernoulli

$$\tilde{z}_e^{(l)} = \text{sigmoid}\left(\frac{1}{t}\log\left(\frac{\pi_l}{1-\pi_l}\right) + \log\left(\frac{u}{1-u}\right)\right)$$

$t =$ temperature, $u =$ sample from $\text{Unif}[0,1]$

  - Solution 2: Augment-REINFORCE-Merge (ARM) gradient estimates
    - Direct optimization of variational parameters
    - Variational loss is unbiased and has low variance
    - More accurate gradient estimate
    - Slightly higher computational complexity

Semi-supervised node classification

TEXAS A&M UNIVERSITY

Method	Cora		Citeseer	
	2 layers	4 layers	2 layers	4 layers
GCN-DO	80.98	78.24	70.44	64.38
GCN-DE	78.36	73.40	70.52	57.14
GCN-DO-DE	80.58	79.20	70.74	64.84
GCN-BBDE	**81.58**	**80.42**	**71.46**	**68.58**
GCN-BBGDC	**81.80**	**82.20**	**71.72**	**70.00**

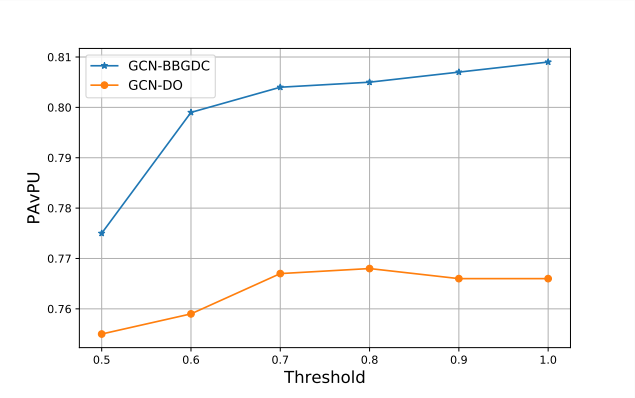
- Learning drop rates improves accuracy
  - More substantial in deeper GCNs
- BBGDC alleviates over-smoothing
  - Specially in deeper GCNs

Method	**Cora** (4 layers)	**Citeseer** (4 layers)
GCN-BDE-ARM	79.95	67.90
GCN-BBDE-ARM	81.78	69.43
GCN-BBGDC-ARM	82.40	70.25

- Beta-Bernoulli better than Bernoulli
  - Due to sparsity imposed by BB
- Direct optimization better than concrete
  - Due to bias in concrete relaxation

Uncertainty quantification

- Patch Accuracy vs Patch Uncertainty (on Cora dataset)
  - $\mathrm{PAvPU} = (n_{ac} + n_{iu})/(n_{ac} + n_{au} + n_{ic} + n_{iu})$
  - Higher $\mathrm{PAvPU} \Rightarrow$ certain predictions are accurate & inaccurate predictions are uncertain

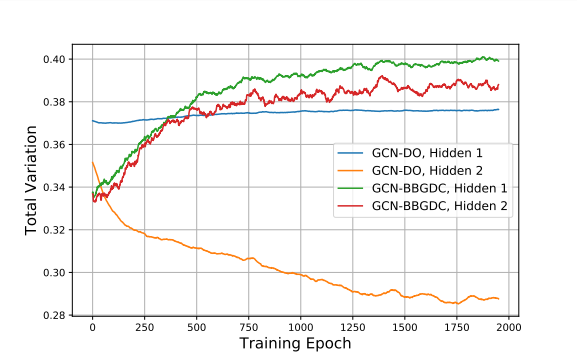


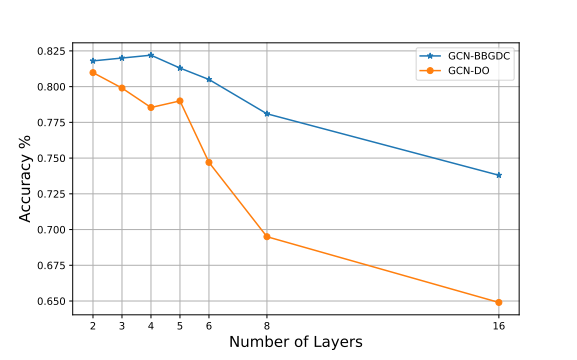
Over-smoothing and over-fitting (Cora)

TEXAS A&M UNIVERSITY

- $\mathrm{TV}(\mathbf{x}) = \|\mathbf{x} - (1/|\lambda_{max}|)\mathbf{A}\,\mathbf{x}\|_2^2$
  - Low TV $\Rightarrow$ most probably over-smoothing occurred

- Over-fitting occurs in deep models
- Over-smoothing intensifies in deeper models





Thanks!

armanihm@tamu.edu
https://github.com/armanihm/GDC