Stochastic block models meet GNN

April, 2021

Table of content

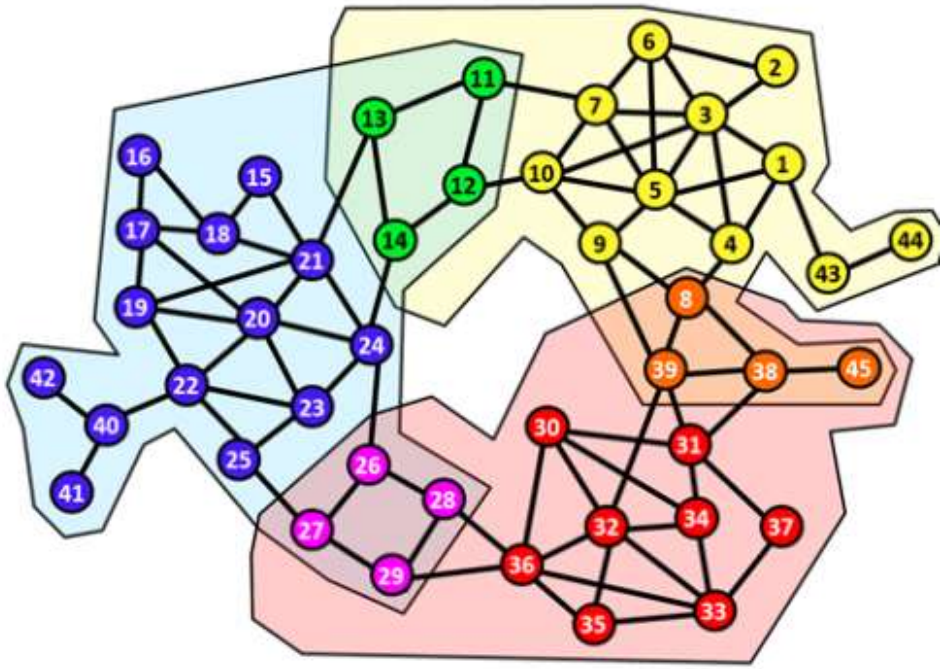
- Intro to community detection
- Intro to stochastic block models (SBM)
- Intro to GNNs:

  refs: Variational Graph Auto-Encoders, semi-supervised classification with GCNs
- Intro to use GNNs to do SBM :

  ref: Stochastic Block models meet Graph Neural Networks

- Two additional topics:
  - SBM's link to clustering
  - Adaptation methods of GNN for featureless graph

Community detection problem



Example of overlapping communities' detection

Important problem!!!
Stereotypical application areas:
Social networks,
Genomics

In fact, if your problem can be represented as graph, you may formulate a community detection problem

Stochastic block models

Definition

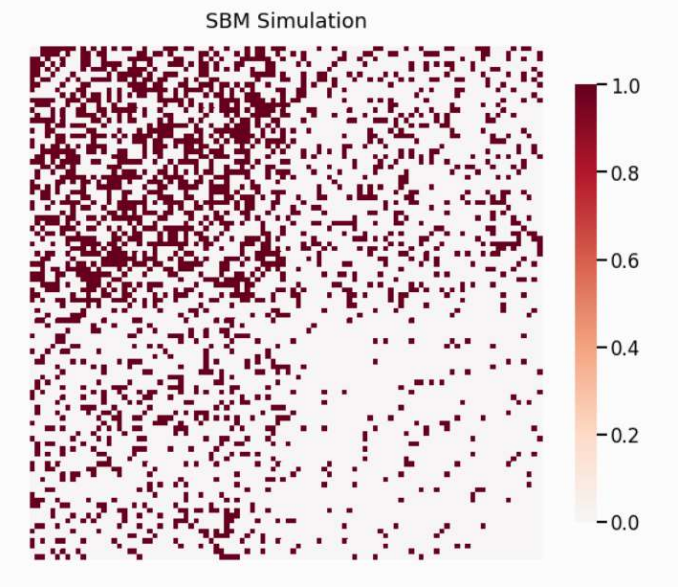
SBM is parametrized by the number of vertices in each community n, and a block probability matrix $P \in \mathbb{R}^{nxn}$ where each element specifies the probability of an edge in a particular block. One can

Example

- P =

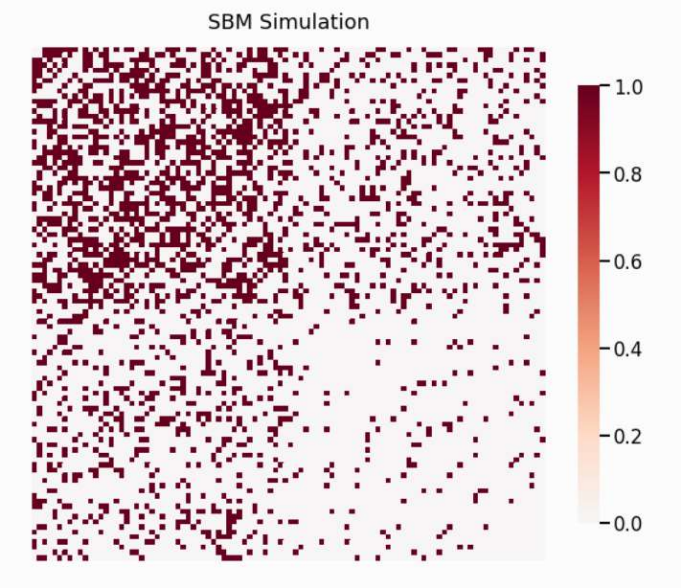
[C_11, C_12

C_21, C_22]

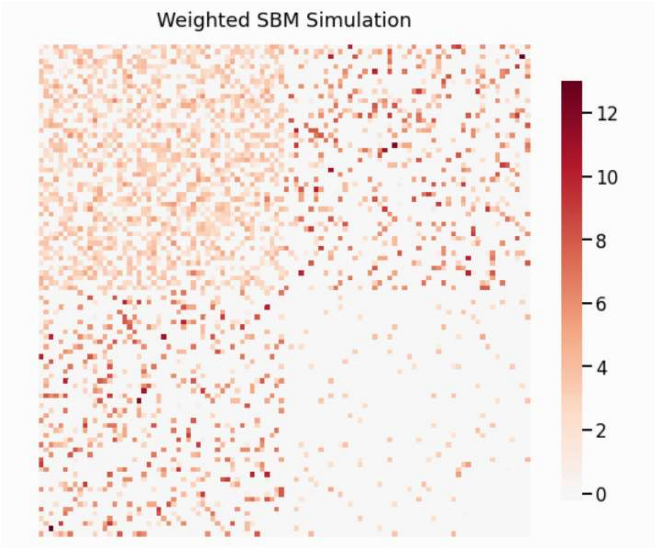


Variations of SBM

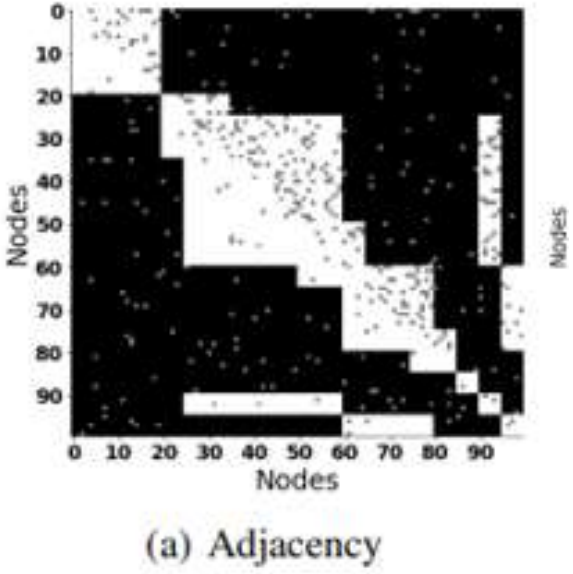
This paper (SBM meets GNN) dealt with this type



Original A\in {0,1}



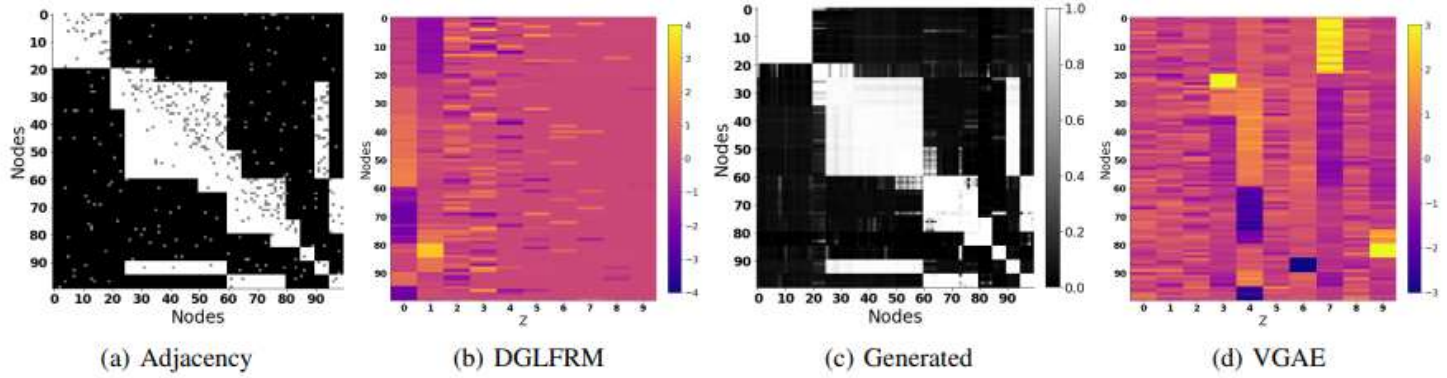
SBM on weighed graph
Weighted A_ij are float



(a) Adjacency

Overlap community

Quick glance of result of Stochastic block models meet GNN



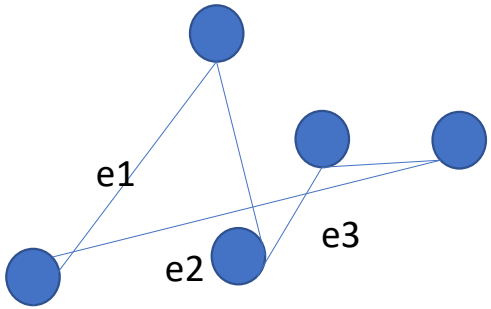
(a) Adjacency (b) DGLFRM (c) Generated (d) VGAE

Figure 2. (a) The synthetic adjacency matrix (where white, black and grey denote link, no-link and hidden parts of the graph). (b) The latent structure of synthetic data learned using DGLFRM. The truncation parameter (K) of the latent structure was fixed to 10. (c) The graph generated by DGLFRM using only the first 2 dimensions of the latent structure($i.e.$ columns 0 and 1 in (b)). The columns 2-9 were all set to zero. The graph is represented as the probability of links; white (black) represents link (no-link) with high probability. (d) The latent structure of synthetic data learned using VGAE.

Graph Neural networks (GNN) vs conventional SBM

Conventional approaches

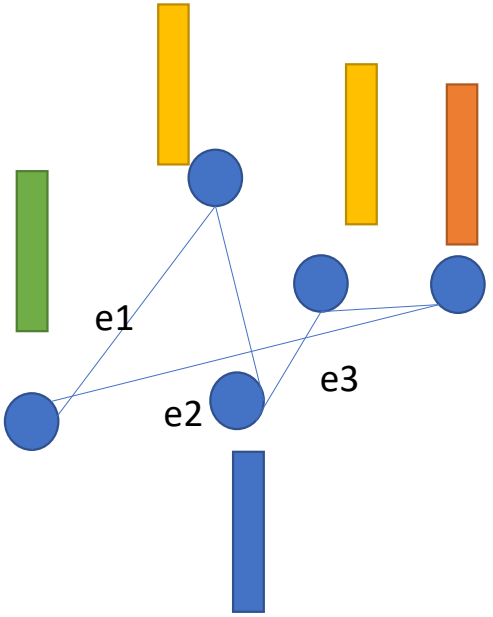
Input (A)



GNN approaches with **Node features**

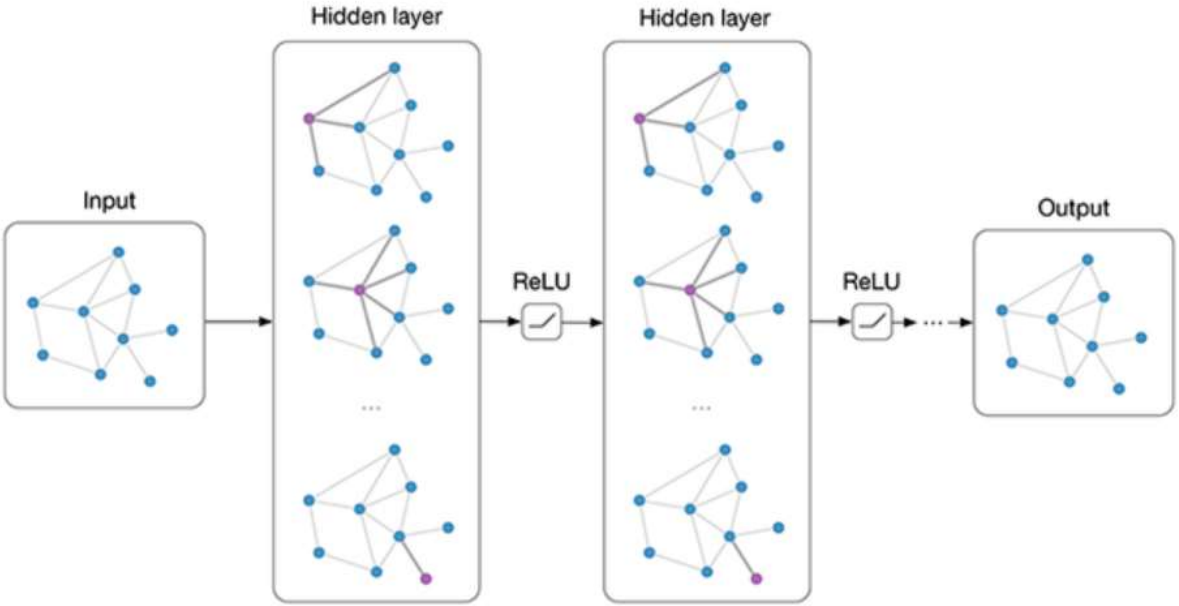
- Features associated with each nodes

Input (A, **X**[feature_vi])



GNN and key references

High level illustration



- Variational Graph Auto-Encoders https://arxiv.org/pdf/1611.07308.pdf
- SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS

https://arxiv.org/pdf/1609.02907.pdf

Technical details

Inference model We take a simple inference model parameterized by a two-layer GCN:

$$q(\mathbf{Z} \,|\, \mathbf{X}, \mathbf{A}) = \prod_{i=1}^{N} q(\mathbf{z}_i \,|\, \mathbf{X}, \mathbf{A}), \quad \text{with} \quad q(\mathbf{z}_i \,|\, \mathbf{X}, \mathbf{A}) = \mathcal{N}(\mathbf{z}_i \,|\, \boldsymbol{\mu}_i, \mathrm{diag}(\boldsymbol{\sigma}_i^2)). \tag{1}$$

Here, $\boldsymbol{\mu} = \mathrm{GCN}_{\boldsymbol{\mu}}(\mathbf{X}, \mathbf{A})$ is the matrix of mean vectors $\boldsymbol{\mu}_i$; similarly $\log \boldsymbol{\sigma} = \mathrm{GCN}_{\boldsymbol{\sigma}}(\mathbf{X}, \mathbf{A})$. The two-layer GCN is defined as $\mathrm{GCN}(\mathbf{X}, \mathbf{A}) = \tilde{\mathbf{A}}\,\mathrm{ReLU}(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}_0)\mathbf{W}_1$, with weight matrices $\mathbf{W}_i$. $\mathrm{GCN}_{\boldsymbol{\mu}}(\mathbf{X}, \mathbf{A})$ and $\mathrm{GCN}_{\boldsymbol{\sigma}}(\mathbf{X}, \mathbf{A})$ share first-layer parameters $\mathbf{W}_0$. $\mathrm{ReLU}(\cdot) = \max(0, \cdot)$ and $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is the symmetrically normalized adjacency matrix.

the form of this propagation rule can be motivated1 via a first-order approximation of localized spectral filters on graphs → details see semi- supervised learning with graph networks

Approximately ~

We consider spectral convolutions on graphs defined as the multiplication of a signal $x \in \mathbb{R}^N$ (a scalar for every node) with a filter $g_\theta = \mathrm{diag}(\theta)$ parameterized by $\theta \in \mathbb{R}^N$ in the Fourier domain, i.e.:

$$g_\theta \star x = U g_\theta U^\top x, \tag{3}$$

where U is the matrix of eigenvectors of the normalized graph Laplacian $L = I_N - D^{-\frac{1}{2}}AD^{-\frac{1}{2}} = U\Lambda U^\top$, with a diagonal matrix of its eigenvalues Λ and $U^\top x$ being the graph Fourier transform of x. We can understand g_θ as a function of the eigenvalues of L, i.e. $g_\theta(\Lambda)$. Evaluating Eq. 3 is

Inference model We take a simple inference model parameterized by a two-layer GCN:

$$q(\mathbf{Z} \,|\, \mathbf{X}, \mathbf{A}) = \prod_{i=1}^{N} q(\mathbf{z}_i \,|\, \mathbf{X}, \mathbf{A}) \,, \quad \text{with} \quad q(\mathbf{z}_i \,|\, \mathbf{X}, \mathbf{A}) = \mathcal{N}(\mathbf{z}_i \,|\, \boldsymbol{\mu}_i, \mathrm{diag}(\boldsymbol{\sigma}_i^2)) \,. \tag{1}$$

Here, $\boldsymbol{\mu} = \mathrm{GCN}_{\boldsymbol{\mu}}(\mathbf{X}, \mathbf{A})$ is the matrix of mean vectors $\boldsymbol{\mu}_i$; similarly $\log \boldsymbol{\sigma} = \mathrm{GCN}_{\boldsymbol{\sigma}}(\mathbf{X}, \mathbf{A})$. The two-layer GCN is defined as $\mathrm{GCN}(\mathbf{X}, \mathbf{A}) = \tilde{\mathbf{A}}\,\mathrm{ReLU}\big(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}_0\big)\mathbf{W}_1$, with weight matrices $\mathbf{W}_i$. $\mathrm{GCN}_{\boldsymbol{\mu}}(\mathbf{X}, \mathbf{A})$ and $\mathrm{GCN}_{\boldsymbol{\sigma}}(\mathbf{X}, \mathbf{A})$ share first-layer parameters $\mathbf{W}_0$. $\mathrm{ReLU}(\cdot) = \max(0, \cdot)$ and $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is the symmetrically normalized adjacency matrix.

Generative model Our generative model is given by an inner product between latent variables:

$$p\left(\mathbf{A} \,|\, \mathbf{Z}\right) = \prod_{i=1}^{N} \prod_{j=1}^{N} p\left(A_{ij} \,|\, \mathbf{z}_i, \mathbf{z}_j\right) \,, \quad \text{with} \quad p\left(A_{ij} = 1 \,|\, \mathbf{z}_i, \mathbf{z}_j\right) = \sigma(\mathbf{z}_i^{\top} \mathbf{z}_j) \,, \tag{2}$$

where A_{ij} are the elements of $\mathbf{A}$ and $\sigma(\cdot)$ is the logistic sigmoid function.

Learning We optimize the variational lower bound $\mathcal{L}$ w.r.t. the variational parameters $\mathbf{W}_i$:

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{Z}|\mathbf{X},\mathbf{A})}\big[\log p\left(\mathbf{A} \,|\, \mathbf{Z}\right)\big] - \mathrm{KL}\big[q(\mathbf{Z} \,|\, \mathbf{X}, \mathbf{A}) \,||\, p(\mathbf{Z})\big] \,, \tag{3}$$

Use GNN for SBM

- Decoder

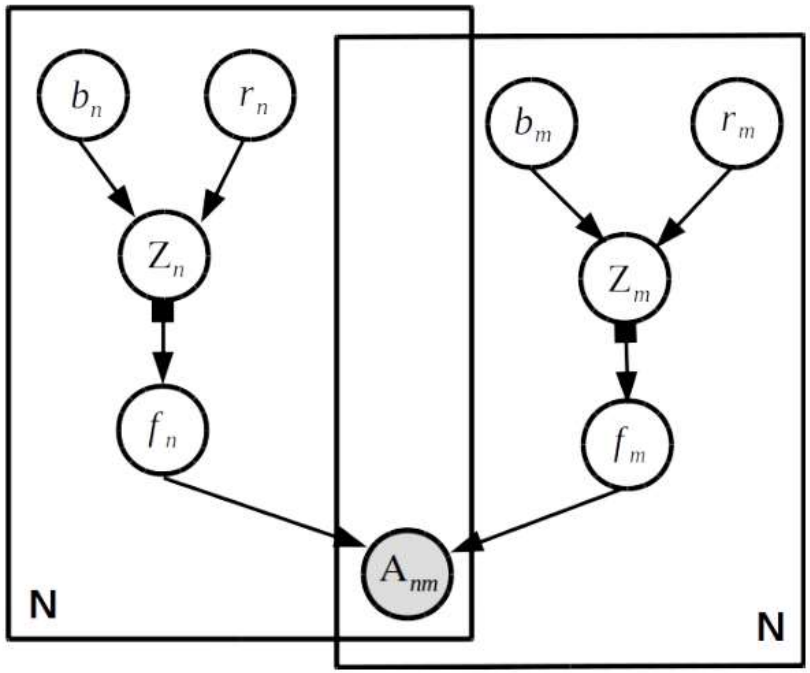
We model the binary vector $\boldsymbol{b}_n \in \{0,1\}^K$, denoting node-community memberships, using the stick-breaking construction of the IBP (Teh et al., 2007), which enables learning of the *effective* K by specifying a sufficiently large truncation level K. The stick-breaking construction is given as follows

$$v_k \sim \text{Beta}(\alpha, 1), \quad k = 1, \ldots, K \tag{2}$$

$$\pi_k = \prod_{j=1}^{k} v_j, \quad b_{nk} \sim \text{Bernoulli}(\pi_k) \tag{3}$$

We further assume a Gaussian prior on membership strengths $\boldsymbol{r}_n \in \mathbb{R}^K$, i.e., $p(\boldsymbol{r}_n) = \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$.

node embeddings $\boldsymbol{z}_n = \boldsymbol{b}_n \odot \boldsymbol{r}_n$



f deterministic nonlinear transformation of the node embeddings zn (this paper uses leakyReLU)

Activation

$$p(A_{nm} = 1 | \boldsymbol{f}_n, \boldsymbol{f}_m) = \sigma(\boldsymbol{f}_n^\top \boldsymbol{f}_m),$$

Use GNN for SBM: Encoder

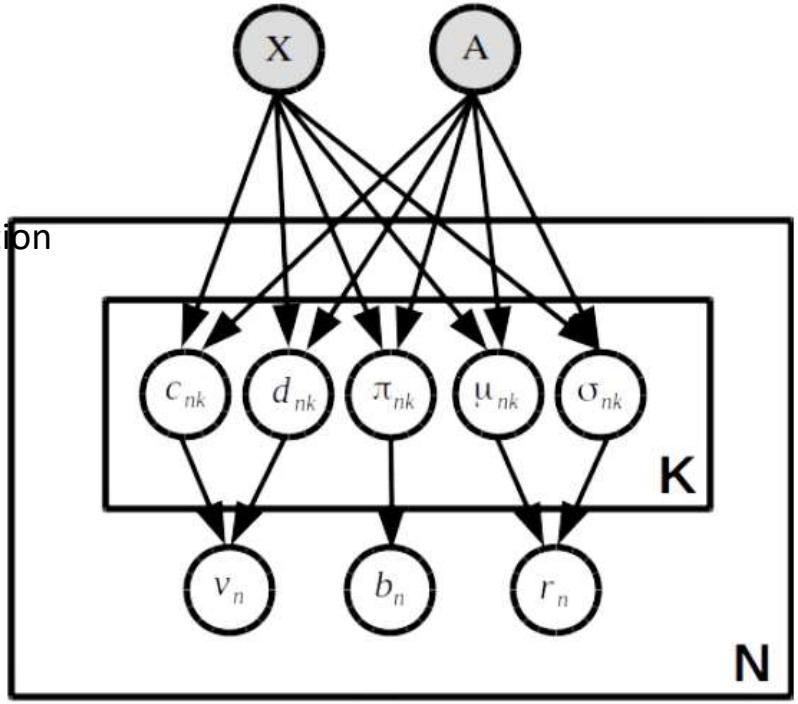
$$q_\phi(v_{nk}) = \text{Beta}(c_{nk}, d_{nk}) \quad k = 1, \ldots, K \quad (4)$$
$$q_\phi(b_{nk}) = \text{Bernoulli}(\pi_{nk}) \quad k = 1, \ldots, K \quad (5)$$
$$q_\phi(\boldsymbol{r}_n) = \mathcal{N}(\boldsymbol{\mu}_n, \text{diag}(\boldsymbol{\sigma}_n^2)) \quad (6)$$

where $c_{nk}, d_{nk}, \pi_{nk}, \boldsymbol{\mu}_n$, and $\boldsymbol{\sigma}_n$ are outputs of a GCN, *i.e.*, $\{c_k, d_k, \pi_k, \mu_k, \sigma_k\}_{n=1}^{n=N} = \text{GCN}(\mathbf{A}, \mathbf{X})$. We use the stochastic gradient variational Bayes (SGVB) algorithm (Kingma & Welling, 2013) to infer the parameters of the variational distributions. Details on reparameterization

Stick-break Construction variables

membership

Amp, and variance



defined by a graph convolutional network that takes as input the network A and node features X (if available) and outputs the parameters of the variational distributions of the model parameter

4. Inference

We define the factorized variational posterior $q_\phi(\boldsymbol{v}, \boldsymbol{b}, \boldsymbol{r})$ as

$$\begin{aligned} q_\phi(v_{nk}) &= \text{Beta}(v_{nk}|c_k(\mathbf{A}, \mathbf{X}), d_k(\mathbf{A}, \mathbf{X})) \\ q_\phi(b_{nk}) &= \text{Bernoulli}(b_{nk}|\pi_k(\mathbf{A}, \mathbf{X})) \\ q_\phi(\boldsymbol{r}_n) &= \mathcal{N}(\boldsymbol{\mu}_n(\mathbf{A}, \mathbf{X}), \text{diag}(\boldsymbol{\sigma}_n^2(\mathbf{A}, \mathbf{X}))) \end{aligned}$$

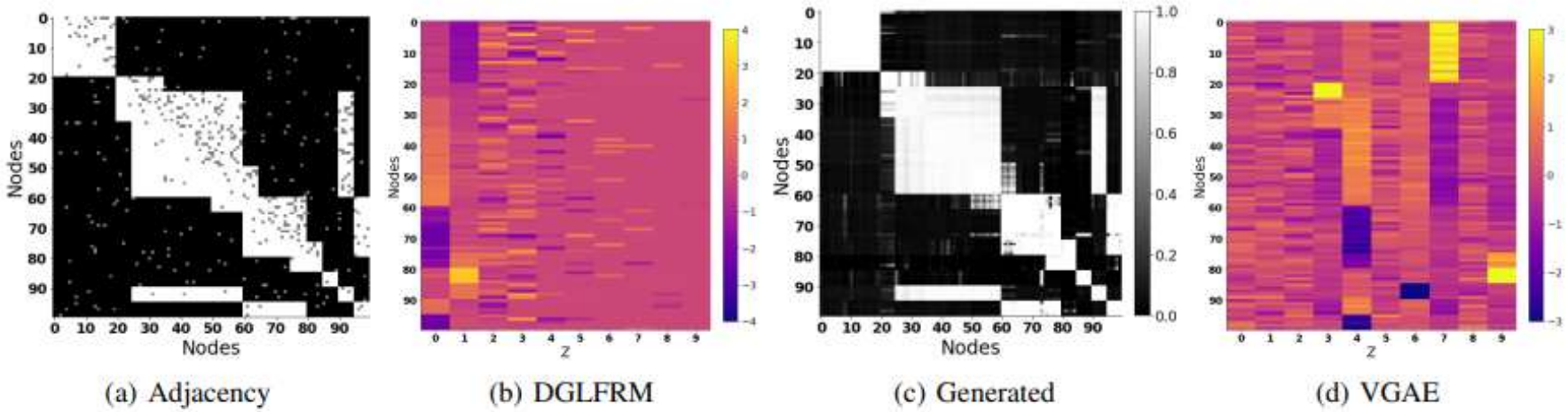
where $c_k, d_k, \pi_k, \boldsymbol{\mu}_n$ and $\boldsymbol{\sigma}_n$ are a function of the GCN encoder, with inputs $\mathbf{A}$ and $\mathbf{X}$. We define the loss function $\mathcal{L}$ parameterized by inference network (encoder) parameters (ϕ) and generator parameters (θ) by minimizing the *negative* of the evidence lower bound (ELBO)

$$\begin{aligned} \mathcal{L} = \sum_{n=1}^{N} \Big(& \text{KL}[q_\phi(\boldsymbol{b}_n|\boldsymbol{v}_n) \,||\, p_\theta(\boldsymbol{b}_n|\boldsymbol{v}_n)] + \text{KL}[q_\phi(\boldsymbol{r}_n) \,||\, p_\theta(\boldsymbol{r}_n)] \\ & + \text{KL}[q_\phi(\boldsymbol{v}_n) \,||\, p(\boldsymbol{v}_n)] \Big) - \sum_{n=1}^{N} \Big(\mathbb{E}_q[\log p_\theta(X_n|\boldsymbol{z}_n)] \Big) \\ & - \sum_{n=1}^{N} \sum_{m=1}^{N} \Big(\mathbb{E}_q[\log p_\theta(A_{nm}|\boldsymbol{z}_n, \boldsymbol{z}_m)] \Big) \end{aligned} \tag{7}$$

where $\text{KL}[q(\cdot)||p(\cdot)]$ is the Kullback-Leibler divergence between $q(\cdot)$ and $p(\cdot)$. Note that here we have also included the loss from the reconstruction of the side information X_n. We have considered that the side information $\mathbf{X}$ and

Training loss

result of Stochastic block models meet GNN



(a) Adjacency (b) DGLFRM (c) Generated (d) VGAE

Figure 2. (a) The synthetic adjacency matrix (where white, black and grey denote link, no-link and hidden parts of the graph). (b) The latent structure of synthetic data learned using DGLFRM. The truncation parameter (K) of the latent structure was fixed to 10. (c) The graph generated by DGLFRM using only the first 2 dimensions of the latent structure($i.e.$ columns 0 and 1 in (b)). The columns 2-9 were all set to zero. The graph is represented as the probability of links; white (black) represents link (no-link) with high probability. (d) The latent structure of synthetic data learned using VGAE.

Node features are useful for additionally information for doing community detection

What if there is no node features ?
- for privacy reasons (no info of user in social network, for example)
- My graph is simply a featureless graph on each node

Simple solution: Replace X with the identity matrix in the GCN.

From Variational Graph Auto-Encoders https://arxiv.org/pdf/1611.07308.pdf

More options you can use - from “On Node Features for Graph Neural Networks”

1 **Centrality-based approaches** - compute a node feature based on its role in the graph**. Usually a scalar**

2.**Learning-based approaches** a node feature is considered as the node **embedding** obtained from an unsupervised learning process which **considers the whole graph structure**

Table 3: Node classification on non-attributed datasets

		BlogCatalog		Wiki		PPI	
		Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
SGC	degree	0.167±0.005	0.027±0.003	**0.176±0.018**	**0.030±0.002**	0.077±0.002	0.026±0.001
	#triangles	0.164±0.001	0.026±0.001	0.139±0.012	0.027±0.003	0.079±0.004	0.029±0.004
	k-core no.	0.161±0.004	0.027±0.002	0.127±0.010	0.023±0.002	**0.090±0.004**	**0.032±0.002**
	egonet no.	0.165±0.009	0.025±0.002	0.155±0.038	0.029±0.005	0.078±0.003	0.024±0.002
	pagerank	**0.168±0.005**	0.026±0.002	0.161±0.019	0.019±0.002	0.077±0.002	0.024±0.002
	coloring no.	0.163±0.005	**0.028±0.002**	0.141±0.014	0.027±0.002	**0.090±0.002**	0.031±0.002
	DeepWalk	0.312±0.008	0.106±0.002	**0.685±0.019**	**0.572±0.028**	**0.256±0.004**	**0.191±0.006**
	HOPE	**0.320±0.005**	**0.118±0.003**	0.636±0.016	0.509±0.014	0.223±0.002	0.143±0.002
None	DeepWalk	**0.385±0.005**	**0.223±0.007**	**0.623±0.011**	**0.522±0.036**	**0.222±0.010**	**0.188±0.003**
	HOPE	0.321±0.003	0.144±0.007	0.597±0.012	0.491±0.021	0.191±0.004	0.145±0.006

- On Node Features for Graph Neural Networks (arxiv.org)

Connection to Clustering

- Clustering algorithms doesn’t model the cross-community connection
- Clustering performance are usually measured by small intra-cluster distance and large inter-cluster distance, which is not necessarily the case for block models
- It is still possible to use clustering algorithm as cheap variant for SBM

Key references

- SBM meet GNN https://arxiv.org/pdf/1905.05738.pdf [main]
- Variational Graph Auto-Encoders https://arxiv.org/pdf/1611.07308.pdf
- semi-supervised classification with GCN https://arxiv.org/abs/1609.02907

(this papers talk more on the case of given one label per community, that are not covered in this talk)

- node features importance https://arxiv.org/ftp/arxiv/papers/1911/1911.08795.pdf