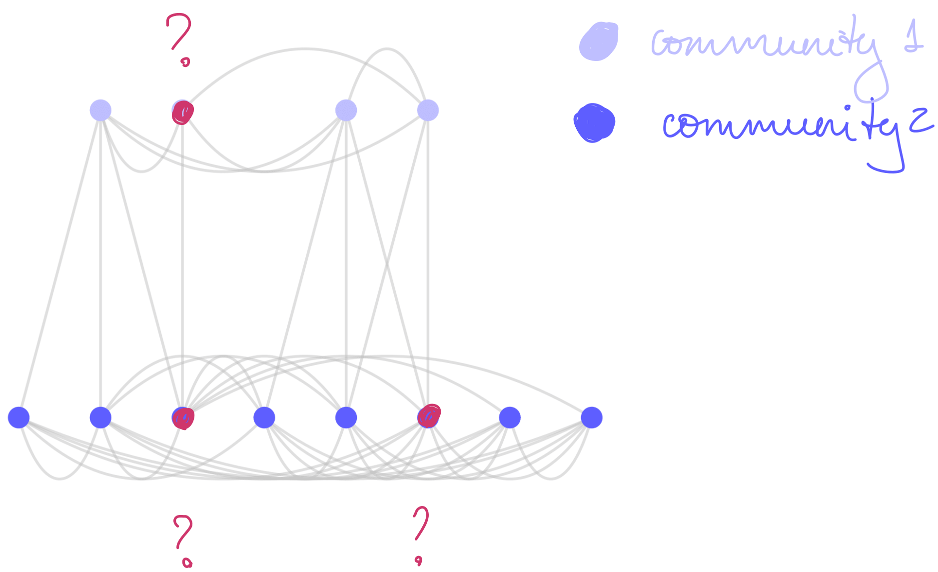
Lecture 7 EN. 553.744 Prof. Luana Ruiz

Today: Community detection:

- info theoretic threshold
- CSBM
- feature-aware spectral emb.
- GNNs
- SPMM



▶ Information-theoretic threshold

→ What happens when $p \approx q$ in the balanced SBM with $c=2$?

When $p = q$, we actually have an Erdős-Rényi graph, in which the edge probability is constant for all nodes; there are no communities to distinguish

But even when $p \neq q$, there is a region around $p = q$ where detection of communities is impossible in an information theoretic sense:

- Let $\text{SNR} = \dfrac{(p-q)^2}{2(p+q)}$ ← degree discrepancy in ≠ subgraphs
  ← $\propto$ average degree ("noise")

- If $\text{SNR} < \frac{1}{n}$, almost exact recovery / detection is impossible; even with infinite time & resources, no algorithm can detect comms in this case.

For the pf, check the works of Massoulié (2014)
Mossel (2014)

E.g.: sparse graphs

$$p = \frac{a}{n} \; ; \quad q = \frac{b}{n} \qquad\qquad \frac{|\mathcal{E}|}{n^2} \to 0 \quad \text{as } n \to \infty$$

$$\text{SNR} = \frac{\left(\frac{a}{n} - \frac{b}{n}\right)^2}{2\left(\frac{a}{n} + \frac{b}{n}\right)} = \frac{(a-b)^2}{n\, 2(a+b)}$$

$$\text{SNR}_{p,q} < \frac{1}{n} \longrightarrow \frac{(a-b)^2}{\cancel{n}2(a+b)} < \frac{1}{\cancel{n}}$$

in this case, comm. detection is impossible

$$\Rightarrow \boxed{\frac{(a-b)^2}{2(a+b)} < 1}$$

▶ Contextual stochastic block model (C-SBM)

An SBM graph, but with node features drawn from a Gaussian mixture:

•) A: $A_{ij} = A_{ji} \sim \text{Ber}(P_{ij})$ $\quad P = \begin{bmatrix} p\cdot 11^T & q\,11^T \\ q\,11^T & p\,11^T \end{bmatrix}$

$(Y \in \{-1, 1\}^n \; ; \; B \in \mathbb{R}^{2\times 2})$

∘) $X_i \sim \sqrt{\frac{\mu}{n}}\, Y_i \cdot u + \frac{Z_i}{\sqrt{d}}$ $\quad u \sim \mathcal{N}(0, I_d/d)$

$Z_i \sim \mathcal{N}(0, I_d)$

$$\boxed{X_i \mid Y_i, u \sim \mathcal{N}\left(\pm\sqrt{\frac{\mu}{n}} \cdot u,\ \frac{I_d}{d}\right)}$$

↳ correction to defn in Lec. 4

▶ Feature-aware spectral embeddings

Let G be a graph w/ adjacency A and node features $X \in \mathbb{R}^{n \times d}$

- Diagonalize $XX^T \in \mathbb{R}^{n \times n}$ as $V' \Lambda' V'^T$
- Pick the top k eigenvectors V'_k
- Define $V_{c+k} = [\, V_c \mid V'_k \,]$ (column wise concatenation)

↳ $\in \mathbb{R}^{n \times (c+k)}$

↳ $f(A, X) = \sigma(V_{c+k} W)$, $\quad W \in \mathbb{R}^{(c+k) \times C}$

↳ In the presence of node features, the information theoretic threshold for community detection becomes

$(p = a/n,\ q = b/n)$

$$\underbrace{\frac{(a-b)^2}{2(a+b)}}_{SNR_A} + \underbrace{\frac{\mu^2 d}{n}}_{SNR_X} > 1 \Rightarrow \text{comm. detection possible in info theoretic sense}$$

⇒ When $p \simeq q$, detection is still possible as long as means are sufficiently separated (high μ and/or high d)

▶ Computational thresholds for spectral clustering (non-rigorous)

$c = 4$

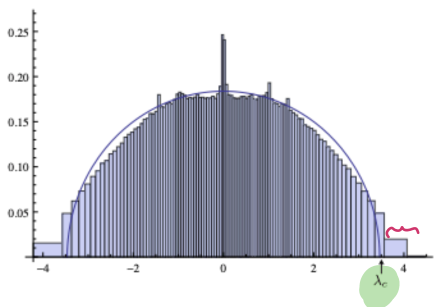


FIG. 1: The spectrum of the adjacency matrix of a sparse network generated by the block model (excluding the zero eigenvalues). Here $n = 4000$, $c_{in} = 5$, and $c_{out} = 1$, and we average over 20 realizations. Even though the eigenvalue $\lambda_c = 3.5$ given by (2) satisfies the threshold condition (1) and lies outside the semicircle of radius $2\sqrt{c} = 3.46$, deviations from the semicircle law cause it to get lost in the bulk, and the eigenvector of the second largest eigenvalue is uncorrelated with the community structure. As a result, spectral algorithms based on A are unable to identify the communities in this case.

↗ Spectral redemption

(figure taken from Krzakala et al. (2013))

↳ in practice, spectral algos. fail above the info theoretic threshold (sometimes significantly above!)

Can GNNs help?

In the example above, having access to $C' > C$ eigenvectors might have helped

↳ not only do we have to fix $\underline{\underline{C}}$ (when unknown), we also have to pick a large enough C so we don't risk missing informative eigenvalues "lost in the bulk"

Back to Lecture 3: Expressivity of graph convolutions

(Thm) If $[\hat{x}]_i \neq 0 \ \forall i$ and $\lambda_i \neq \lambda_j \ \forall i \neq j$, there always exist $K \leq n$ coefficients $h_0, h_1, \dots, h_{K-1}$ s.t. $y = H(S) \cdot x$

Let y be our community assignment vector;
as long as $[\hat{x}]_i \neq 0 \ \forall i$ & $\lambda_i \neq \lambda_j \ \forall i, j$, $\exists$ a graph conv. (1-layer linear GNN) that approximate y.

motivates using GNNs for (semi-supervised) comm. detection on graphs.

I.e., we solve:

$$\min_{f} \sum_{i \in \mathcal{T}} \ell\left([f(A,X)]_i , Y_i \right)$$

surrogate of 0-1 loss

some parametric fcn of A, X

one-hot comm. vector of node i, $[Y_{ic}] = 1$ if i in c

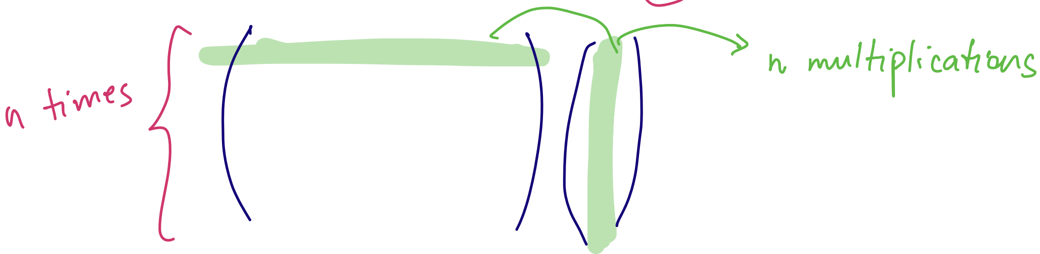
where $f(A,X) = \Phi_{\mathcal{H}}(A,X)$ → GNN w/ learnable parameters $\mathcal{H}$

(see Lec. 5)

A note on sparse matrix-vector multiplications:

$\boxed{Sx} \quad S^{k-1}x = S^{k-2}\underline{(Sx)}$

The usual matmul operation in PyTorch requires $O(n^2)$ to compute Sx: $[Sx]_i = \sum_{j=1}^{n} [A_{ij}]\, x_j$ for $1 \leq i \leq n$



Typically the GSO S (i.e., A, L, etc.) is a sparse $n \times n$ matrix. I.e., $\frac{|\mathcal{E}|}{n^2} << 1$

↳ $|\mathcal{E}|$ nonzero entries $\Rightarrow$ $O(|\mathcal{E}|)$ complexity possible

But: to achieve it, need to use sparse tensor (of S) representation & sparse matrix multiplication ($\neq$ operations in PyTorch)

How sparse matmuls work

→ Coordinate (COO) representation

$$A = \begin{bmatrix} 12 & 0 & 26 & 0 \\ 0 & 0 & 19 & 14 \\ 26 & 19 & 0 & 0 \\ 0 & 14 & 0 & 7 \end{bmatrix}$$

becomes:

	row	col.	value
(	0,	0,	12)
(	0,	2,	26)
(	1,	2,	19)
	⋮		

can we curb index repetition?

→ Compressed sparse row (CSR) representation
↳ most popular!

pointer (i)

0	1	2	3	
0	2			

column (k)

0	1	2	3	
0	2	2	3	

value

0	1				
12	26	19	14		

for each row (i):

for (k): pointer$[i]$ to $\underbrace{\text{pointer}[i+1]-1}_{\text{included}}$:

$y[i] = y[i] + \text{value}[k] \cdot x[\text{column}[k]]$

total operations: $\sum_{i=1}^{n} |N_i| = \sum_{i=1}^{n} d_i = \mathbb{1}^T A \mathbb{1} = |\mathcal{E}|$

In PyTorch: ~~ta~~ transform to a sparse tensor:

S.to_sparse() or S.to_sparse_csr()

(COO)

matrix vector mult. as before: S @ x

or torch.mv(S, x)

for both COO
CSR

Feel free to use these in your HW!