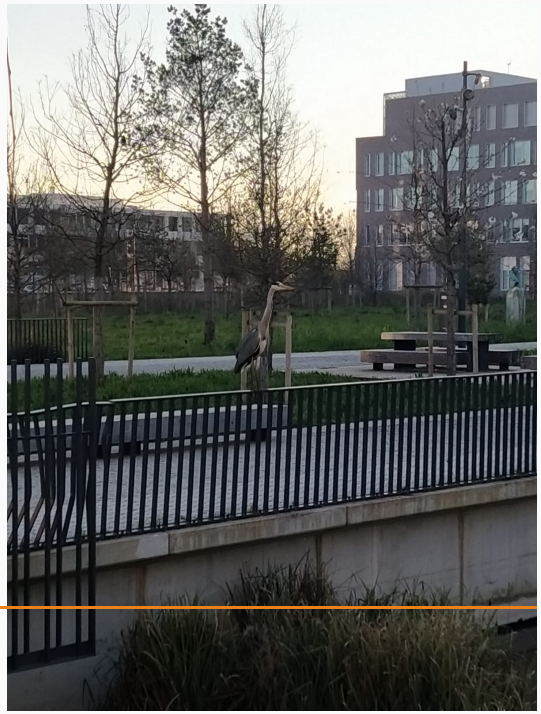
Graph Neural Network: some stuff



Pierre BARBILLON (with a little help from Emre Anakök)

Rochebrune Top, Spring 2024

Outline

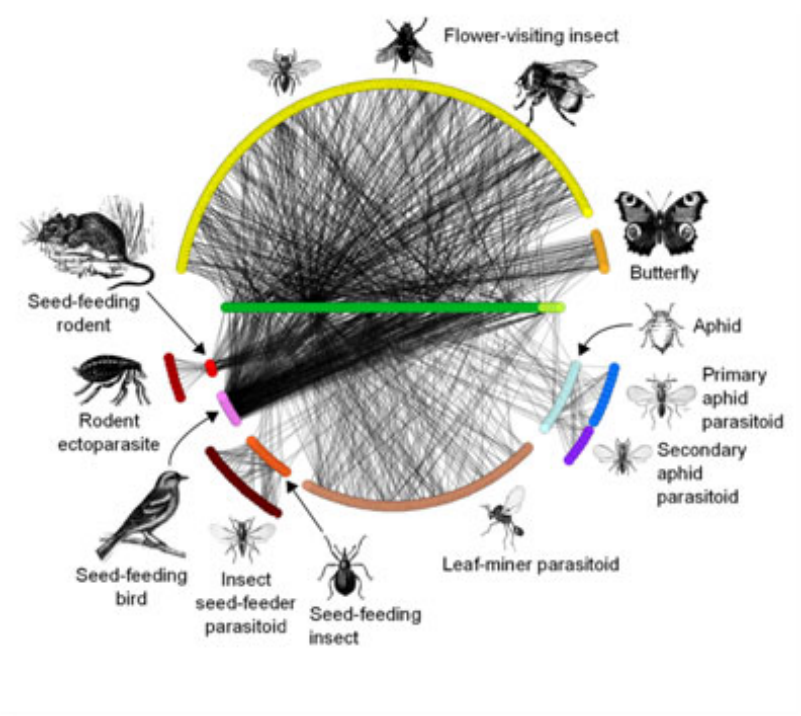
Outline

Sociology

- **Nodes**: individuals or organizations
- **Edges**: advice, competition, ...
- **Examples of objectives**: characterizing the role of individuals in the network, link their role to covariates

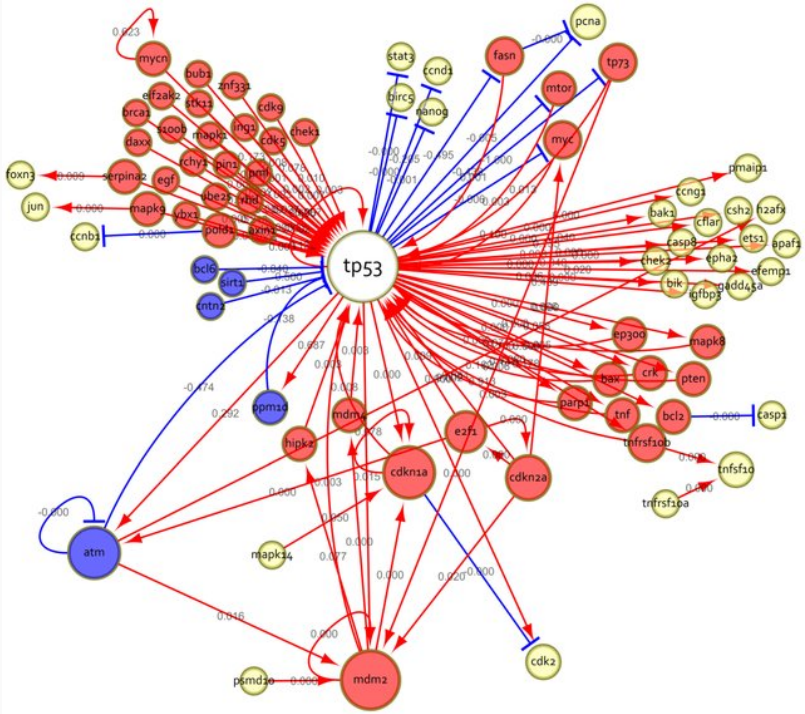
Ecology

- **Nodes:** species (plants or animals)
- **Edges:** predation, pollination, competition...
- **Examples of objectives** : characterizing the structure of the network because it conditions their robustness to the disappearance of species.



Biology

- **Nodes:** genes, metabolites, proteines,
- **Edges:** Regulation, co-expression, reactions,
- **Examples of objectives:** Determine groups of genes co-expressed together under some stresses.



Networks

Graph $G = (V, E, W)$ with

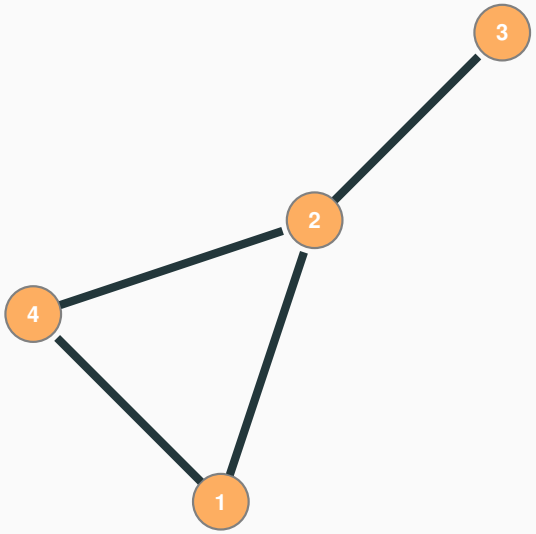
- a set of nodes $V = \{1, \ldots, N\}$,
- a set of edges $E \subset V^2$, particular cases: (un)directed, with(out) loop,...
- additional information on edges, $w \in W$ containing weights (number of interactions, positive or negative interaction,...)

Attributes of:

- **nodes**, for any $i \in V$, X_i attributes of a node (taxon, gender, age, social group,...), or information derived from the edges: degree of i,
- **edges**, for any $e = (i, j) \in E$, the edges may have an attribute coming from the two nodes (difference of ages, same gender...,) or particular attribute (date of interaction,...)
- **network**, global attribute derived from the edges mean connectivity, diameter, or an associated variable.

Network encoding/representation

Simple network



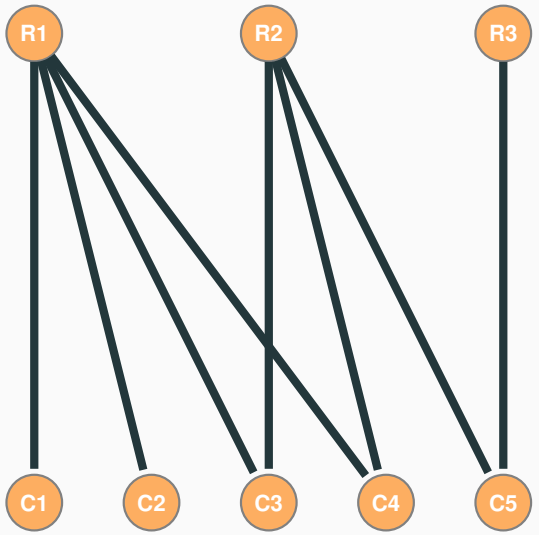
Adjacency matrix:

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

edge list:

E={(1,2),(2,3),(1,4),(2,4)}

Bipartite network



Incidence matrix:

$$B = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

edge list:

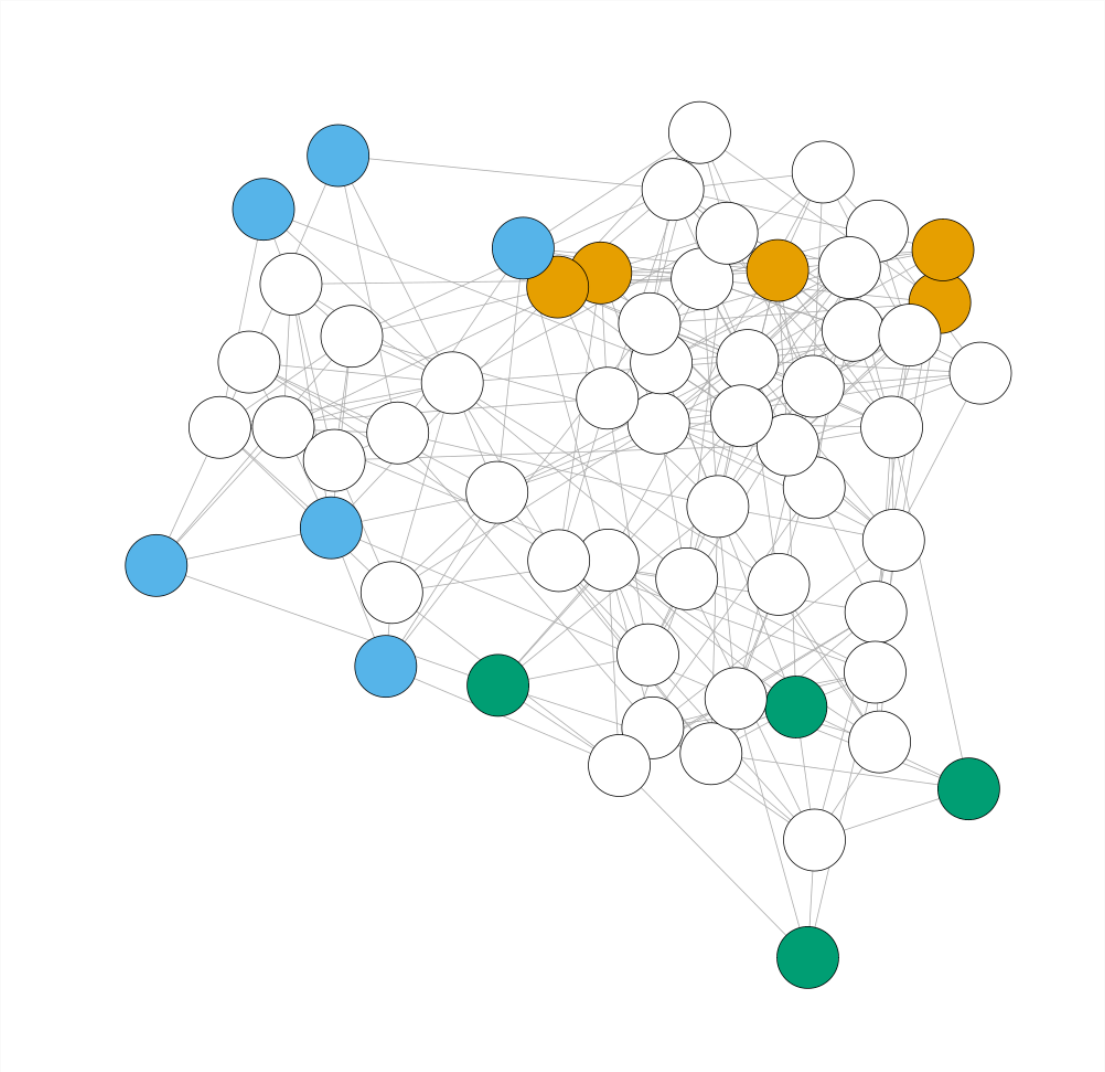
$E = \{(R1, C1), (R1, C2), (R1, C3)...\}$

Outline

Semi-supervised learning on nodes

Data: $G = (V, E)$ and labels in $\{1, \ldots, K\}$ for a subset of V,

- learn $f : i \in V \mapsto \{1, \ldots, K\}$,
- leverage the network structure E.



Clustering of nodes in a graph / Embedding

Data: a graph $G = (V, E)$.

Goal:

- Partition on V.
- Embedding: latent representation of nodes in $\mathbb{R}^d$: $f : i \in V \mapsto \mathbb{R}^d$.

Classification of graphs or regression on graphs

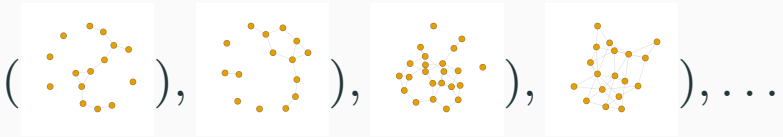
Data:

$$(\ \text{  }\ , y_1),\ \text{  }\ , y_2),\ \text{  }\ , y_3),\ \text{  }\ , y_4), \ldots$$

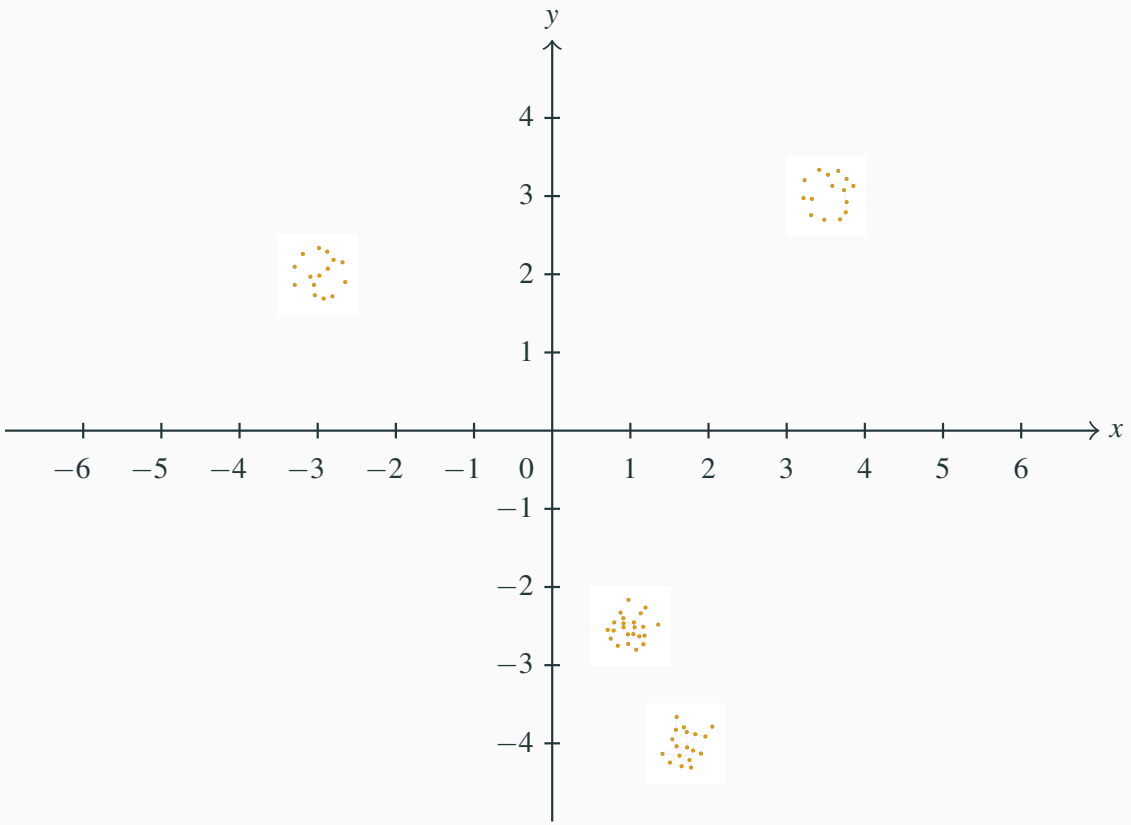
Goal: learn $f : G = (V, E) \mapsto y \in \{1, \ldots, K\}$ or $f : G = (V, E) \mapsto y \in \mathbb{R}$.

Clustering of graphs / Embeddings

Data:



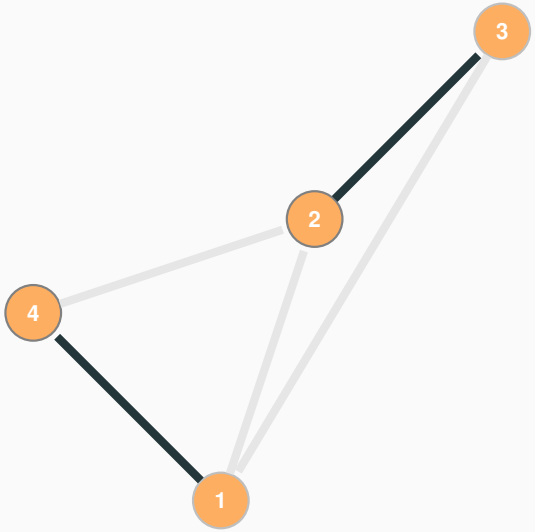
Goal: learn a partition of graphs , learn an embedding:



Predict of dyads, missing links

Data: a graph G with missing or incomplete data.

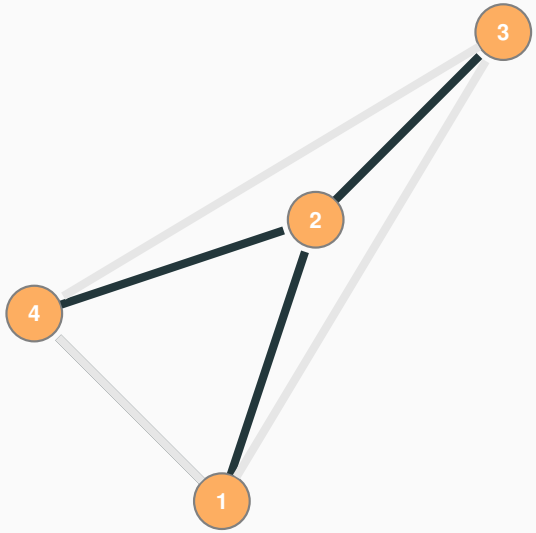
missing data



Adjacency matrix:

$$A = \begin{pmatrix} 0 & \texttt{NA} & \texttt{NA} & 1 \\ \texttt{NA} & 0 & 1 & \texttt{NA} \\ \texttt{NA} & 1 & 0 & 0 \\ 1 & \texttt{NA} & 0 & 0 \end{pmatrix}$$

incomplete data



Adjacency matrix:

$$A = \begin{pmatrix} 0 & 1 & \texttt{NA} & \texttt{NA} \\ 1 & 0 & 1 & 1 \\ \texttt{NA} & 1 & 0 & \texttt{NA} \\ \texttt{NA} & 1 & \texttt{NA} & 0 \end{pmatrix}$$

Goal: Predict NA to $\{0, 1\}$ or predict most likely existing links.

Outline

Models with latent variables

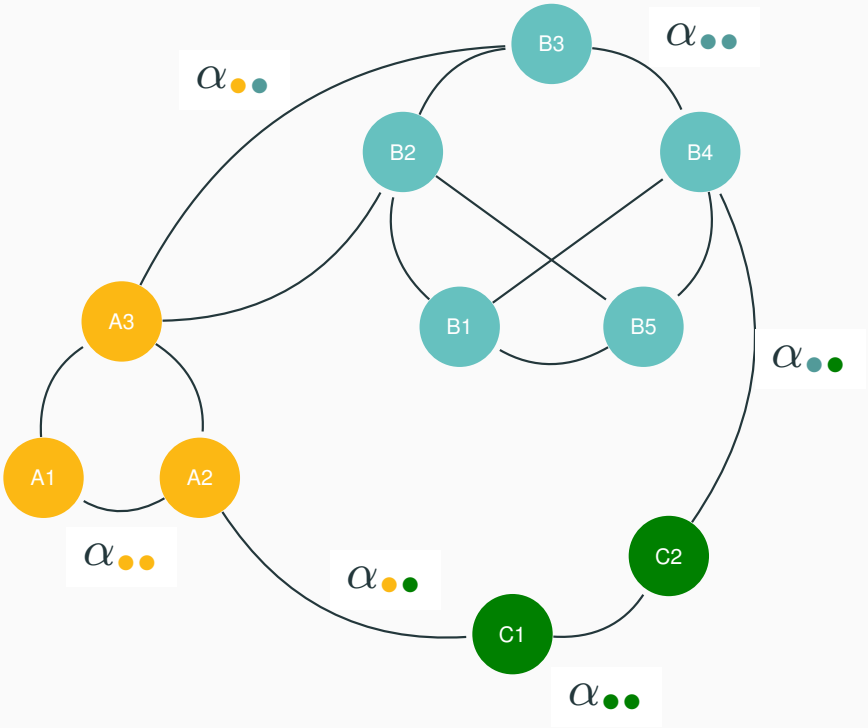
Review: [Matias and Robin, 2014]

- $\boldsymbol{Z} = (Z_1, \ldots, Z_N)$ independent latent variables in $\{1, \ldots, K\}$ or in $\mathbb{R}^d$,
- $Y_{ij}|Z_i, Z_j \overset{ind}{\sim} \mathcal{F}(\alpha_{Z_i,Z_j})$ for all dyads (i, j).
- can include covariates: $Y_{ij}|Z_i, Z_j \overset{ind}{\sim} \mathcal{F}(\alpha_{Z_i,Z_j}, x_{i,j})$.
- e.g. $Y_{ij}|Z_i, Z_j \overset{ind}{\sim} b\Big(1/\big(1 + \exp(-\alpha_{Z_i,Z_j} + \beta^\top x_{i,j})\big)\Big)$.

Two classical families:

- if Zs are categorical $\rightarrow$ Stochastic Block Models [Nowicki and Snijders, 2001],
- if Zs are in a continuous space $\rightarrow$ Latent space models [Peter D Hoff and Handcock, 2002]

Stochastic Block Model : illustration



Parameters

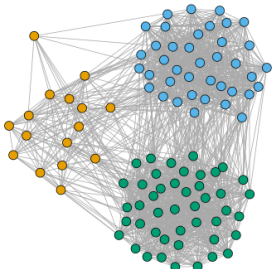
Let N nodes divided into 3 clusters

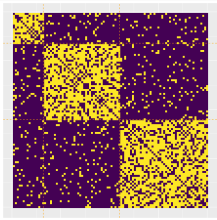
- $\{\bullet, \bullet, \bullet\}$ clusters
- $\pi_\bullet = \mathbb{P}(i \in \bullet),\ i = 1, \ldots, N$
- $\alpha_{\bullet\bullet} = \mathbb{P}(i \leftrightarrow j | i \in \bullet, j \in \bullet)$

$$\mathbf{Y} \sim \mathrm{SBM}_N(Q, \boldsymbol{\pi}, \boldsymbol{\alpha})\,.$$

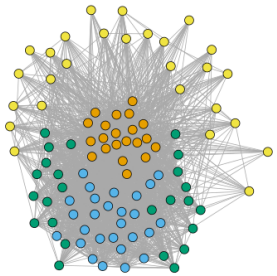
Simulations under the SBM

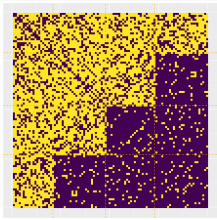
$$\alpha = \begin{pmatrix} 0.70 & 0.09 & 0.09 \\ 0.09 & 0.70 & 0.09 \\ 0.09 & 0.09 & 0.70 \end{pmatrix}$$



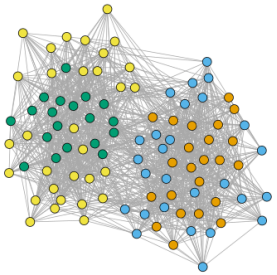


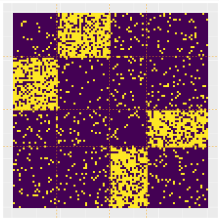
$$\alpha = \begin{pmatrix} 0.70 & 0.70 & 0.70 & 0.70 \\ 0.70 & 0.70 & 0.70 & 0.09 \\ 0.70 & 0.70 & 0.09 & 0.09 \\ 0.70 & 0.09 & 0.09 & 0.09 \end{pmatrix}$$





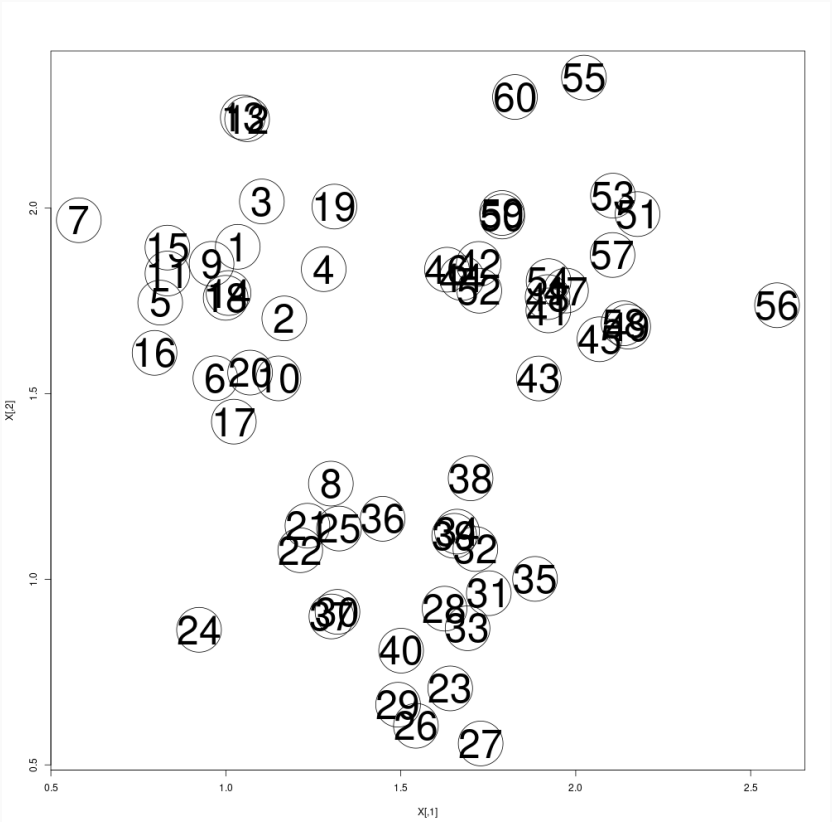
$$\alpha = \begin{pmatrix} 0.09 & 0.70 & 0.09 & 0.09 \\ 0.70 & 0.09 & 0.09 & 0.09 \\ 0.09 & 0.09 & 0.09 & 0.70 \\ 0.09 & 0.09 & 0.70 & 0.09 \end{pmatrix}$$

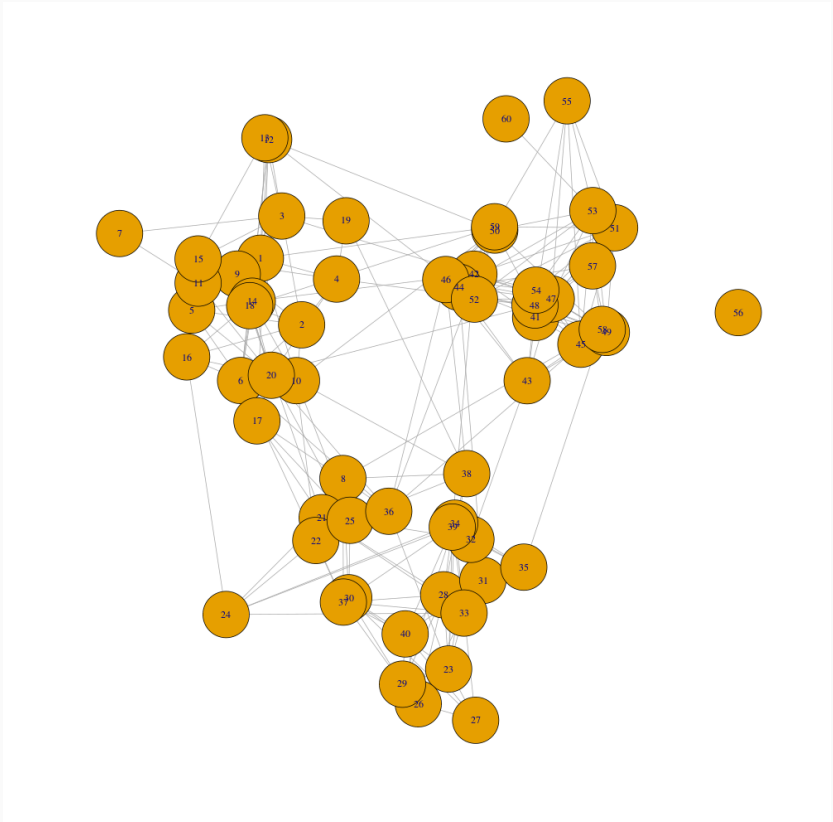




Latent space model

- $\forall i \in \{1, \ldots, N\}$, $Z_i \overset{ind}{\sim}$ Mixture$\mathcal{N}((\mu_k)_k, (\Sigma_k)_k)$,
- $\forall (i,j)$,$Y_{ij}|Z_i, Z_j \overset{ind}{\sim} b(\exp(-\|Z_i - Z_j\|/\sigma^2))$.





(Generalised) random dot product graph

Alternative to the distance between latent positions, the dot product can be used:

$$\forall (i,j),\ Y_{ij} | Z_i, Z_j \overset{ind}{\sim} b(Z_i \cdot Z_j = Z_i^\top Z_j).$$

[Rubin-Delanchy et al., 2022] proposed a generalisation:

$$\forall (i,j),\ Y_{ij} | Z_i, Z_j \overset{ind}{\sim} b(Z_i I_{p,q} Z_j)$$

with

$$I_{p,q} \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix}.$$

Outline

Laplacian of a graph

For $G = (V, E)$ an undirected graph s.t. $V = \{1, \ldots, N\}$ and A the corresponding adjacency matrix.

- Degree of a vertex/node: $d_i = \sum_j A_{ij}$,
- Unnormalized Laplacian: $L = D - A$ with $D = \text{diag}(d_1, \ldots, d_N)$,

Properties:

- for $x \in \mathbb{R}^n$, $x^\top L x = \frac{1}{2} \sum_j^N A_{ij}(x_i - x_j)^2$,
- L is symmetric and positive definite,
- the smallest eigenvalue is 0 and associated with the vector $\mathbb{1}$,
- the order of multiplicity of 0 is the number of connected components.

[Von Luxburg, 2007]

Normalized Laplacians:

$$
\begin{aligned}
L_{sym} &= D^{-1/2} L D^{-1/2} = I_N - D^{-1/2} A D^{-1/2} \\
L_{rw} &= D^{-1} L = I_N - D^{-1} A
\end{aligned}
$$

Properties:

- for $x \in \mathbb{R}^n$, $x^\top L_{sym} x = \frac{1}{2} \sum_j^N A_{ij} (x_i/\sqrt{d_i} - x_j/\sqrt{d_j})^2$,
- L_{sym} and L_{rw} are symmetric and positive definite,
- the smallest eigenvalue is 0,
- the order of multiplicity of 0 is the number of connected components.

Spectral Clustering:

Input: Adjacency Matrix $A \in \mathbb{R}^{N \times N}$, number k of clusters to construct.

- Compute the unnormalized Laplacian L.
- Compute the first k eigenvectors $u_1, \ldots, u_k$ of L.
- Let $U \in \mathbb{R}^{N \times k}$ be the matrix containing the vectors $u_1, \ldots, u_k$ as columns.
- For $i = 1, \ldots, N$, let $z_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of U.
- Cluster the points $(z_i)_{i=1,\ldots,N}$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \ldots, C_k$.

Random walks point of view

Transition from vertex v_i to vertex v_j given by $p_{ij} := \frac{A_{ij}}{d_i}$.

Transition matrix:

$$P = (p_{ij})_{i,j=1,\dots,n}, \quad P = D^{-1}W.$$

Stationary distribution: if G is connected, unique stationary distribution $\pi = (\pi_1, \dots, \pi_n)^T$ with $\pi_i = \frac{d_i}{\mathrm{vol}(V)}$.

Relation with Laplacian: $L_{\mathrm{rw}} = I - P \Rightarrow$ same eigenvectors.

Node2vec: proposes an embedding from random walks on graph [Grover and Leskovec, 2016].

Diffusion point of view

Heat equation on graph:

- S given subset of V with fixed temperature,
- heat exchanges according to (for $i \notin S$):

$$\frac{dT_i}{dt} = \sum_i^n A_{ij}(T_j - T_i) = -(LT)_i \,.$$

- Equilibrium: Laplace equation when $(LT)_i = 0$ or with RW $T_i = (PT)_i$.

[Bonald and De Lara, 2023] relies on this to semi-supervised the graph.

Outline

Convolution on graphs

- particular structure,
- isomorphism of graphs up to relabelling the nodes,
- large graphs but sparse,
- convolution on graph, convolution on images (images can be seen as graph with fixed number of neighbors)

Convolution with neighbors: x features on nodes:

$$h_i = \sum_{j \in \mathcal{N}(i)} x_j$$

$\mathcal{N}(i)$ is the set of neighbors of node i.

Polynomial filters

- Polynomial

$$p_w(L) = w_0 + w_1 L + w_2 L^2 + \ldots + w_d L^d = \sum_{r=0}^{d} w_r L^r.$$

- Convolution of node feature $\mathbf{x}$:

$$h = p_w(L)\mathbf{x}.$$

- if $p_w(L) = 1$, $\mathbf{h} = p_w(L)\mathbf{x} = w_0 I\mathbf{x} = \mathbf{x}$,
- if $p_w(L) = L$, $h_i = (L\mathbf{x})_i = \sum_j (D_{ij} - A_{ij})x_j = D_i x_i - \sum_{j \in \mathcal{N}_i} x_j$,
- $\text{dist}_G(i,j) > r \implies L^r_{vu} = 0$,
- $h_i = (p_w(L)\mathbf{x})_i = (p_w(L))_i\mathbf{x} = \sum_{r=0}^{d} w_r (L^r\mathbf{x})_i = \sum_{r=0}^{d} w_r \sum_j L^r_{ij} x_j = \sum_{r=0}^{d} w_r \sum_{j,\text{dist}_G(j,i) \leq r} L^r_{ij} x_j$.
- independent of the ordering of the node.

[Defferrard et al., 2016]

Convolution with polynomial filters

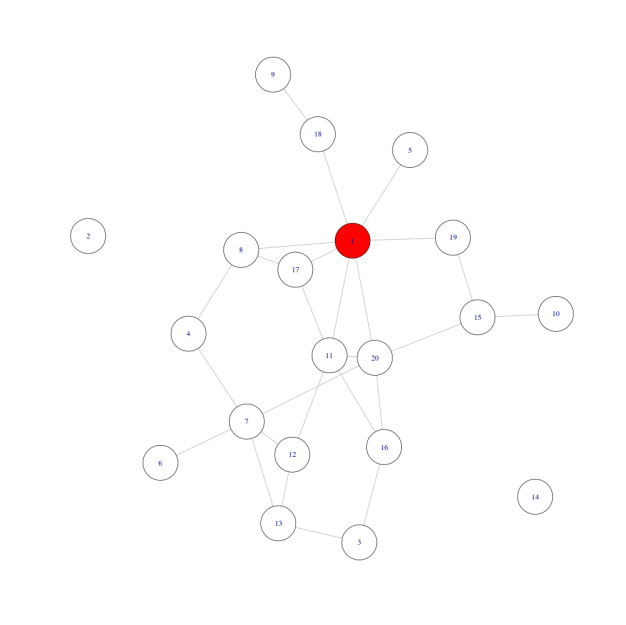
If we have K polynomial filters $p_{w^{(k)}}(L)$ with $w^{(k)}$ trainable parameters.

- $h^{(0)} = x$,
- iterate from $k = 1, \ldots$
  - compute $p^{(k)} = p_{w^{(k)}}(L)$,
  - Matrix computation: $g^{(k)} = p^{(k)} \cdot h^{(k-1)}$,
  - non linear function: $h^{(k)} = \sigma(g^{(k)})$.

If we use $p_{w^{(k)}}(L) = L$:

- $$h_i = (L\mathbf{x})_i = \sum_j (D_{ij} - A_{ij})x_j = D_i x_i - \sum_{j \in \mathcal{N}_i} x_j$$
- we aggregate over immediate neighbors,
- and we combine with the node feature,
- the aggregation is node-order equivariant $\Rightarrow$ overall convolution is node-order equivariant,
- convolutions can be thought of as 'message-passing' between adjacent nodes,
- repeating 1-hop localized convolutions K times makes convolution effective K hops away.

convolutional layer



Graph Convolution Networks

[Kipf and Welling, 2016]

$$h_i^{(\ell+1)} = \sigma\left(\mathbf{W}^{(\ell+1)} \sum_{j \in \mathcal{N}(i) \cup \{i\}} \frac{1}{c_{i,j}} \cdot \mathbf{x}_j^{(\ell)}\right)$$

Importance of normalization $c_{i,j}$.

Matrix form

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l)}W^{(l)}\right)$$

with

- $W^{(l)}$ a matrix of trainable parameters,
- $\tilde{A} = A + I$,
- D the diagonal matrix of degrees of $\tilde{A}$.

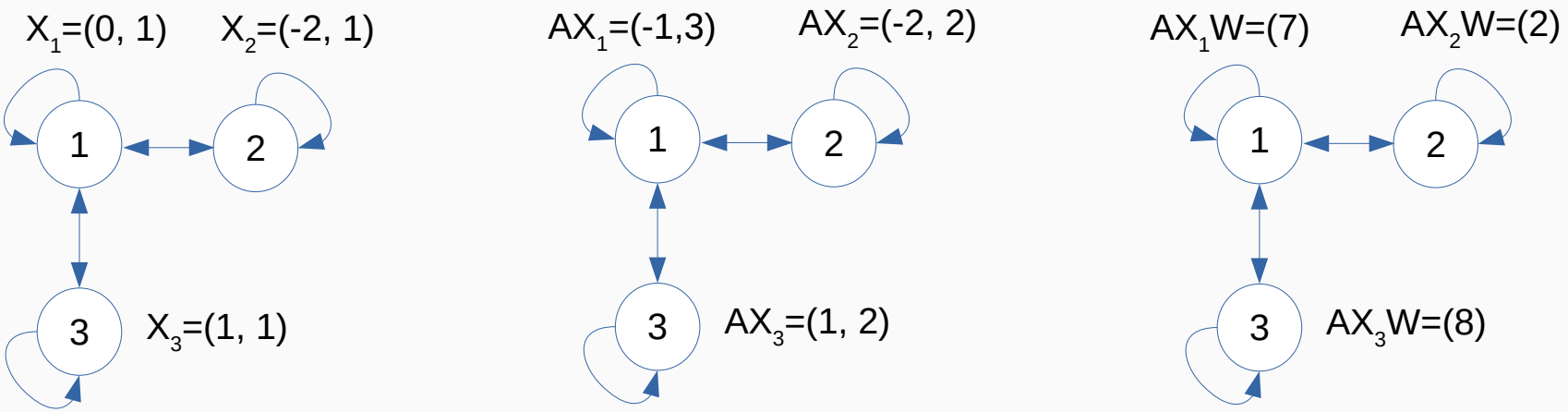
first-order approximation of localized spectral filters proposed in [Defferrard et al., 2016]

Importance of normalization

Different choices:

- No normalization $\tilde{A}$
  - $h_i^{l+1} = \sum_{j, j \in \mathcal{N}(i)} A_{ij} h_j^l$,
  - Eigenvalue of $\tilde{A}$ larger than $1 \Rightarrow$ exploding largest eigenvalue when stacking layers,
- row normalization $A_{\text{row}} = D^{-1} A$,
  - $h_i^{l+1} = \sum_{j, j \in \mathcal{N}(i)} A_{ij} \frac{h_j^l}{d_i}$
  - largest eigenvalue is 1 but not taken into account connectivity of neighbors,
- col normalization $A_{\text{col}} = A D^{-1}$
  - $h_i^{l+1} = \sum_{j, j \in \mathcal{N}(i)} A_{ij} \frac{h_j^l}{d_j}$
  - largest eigenvalue is 1 but put too much weight on well connected nodes,
- Naive normalization $A_{\text{naive}} = D^{-1} A D^{-1}$
  - $h_i^{l+1} = \sum_{j, j \in \mathcal{N}(i)} A_{ij} \frac{h_j^l}{d_j d_i}$
  - largest eigenvalue is < 1 and vanishes when stacking layers,
- symmetric normalization $A_{\text{sym}} = D^{-1/2} A D^{-1/2}$
  - $h_i^{l+1} = \sum_{j, j \in \mathcal{N}(i)} A_{ij} \frac{h_j^l}{\sqrt{d_j d_i}}$
  - largest eigenvalue is 1, combine row and col normalization.

Graph Convolutional Network



$$\tilde{A} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}, X = \begin{pmatrix} 0 & 1 \\ -2 & 1 \\ 1 & 1 \end{pmatrix}, W = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

Other kinds of GNN

- Graph Convolution Networks as we have seen,
- Graph Attention Networks (GAT) [Casanova et al., 2018],
  - $h_i^l = \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha^l(i,j) W h_j^{l-1}\right)$,
  - $\alpha^l(i,j)$ is the attention function,
  - $\alpha^l(i,j) = \text{softmax}\left(\sigma'\left(a^\top \cdot (Wh_i, Wh_j)\right)\right)$.
- Graph SAGE (SAmple and agGrEgate) [Hamilton et al., 2017],
  - $h_{\mathcal{N}(i)}^l = AGGREGATE_k(\{h_j^{l-1}, j \in \mathcal{N}(i)\})$,
  - $h_i^l = \sigma(W^l \cdot CONCAT(h_i^{l-1}, h_{\mathcal{N}(i)}^l)$,
  - $h_i^l = h_i^l / \|h_i^l\|$.
- Graph Isomorphism Network (GIN) [Xu et al., 2018].

see https://distill.pub/2021/understanding-gnns/

Outline

Semi-supervised learning on nodes

Data: $G = (V, E)$ a network with N nodes, $m\%$ of nodes with an observed labels in $\{1, \ldots, Q\}$, V set of edges is known, (features on nodes X).

Goal: Classify nodes without labels.

Architecture:

- X can be a vector of the degrees of nodes, a number for each node, or an identity matrix...
- 2 or 3 GCN layers with given numbers of features,
- Last layer is a linear transformation in a K dimension space : for each p point from the dataset $(h^L_{p1}, \ldots, h^L_{pK})$.

Loss: Cross entropy:

$$loss(x, y) = \frac{1}{n_{\text{train}}} \sum_{p=1}^{n_{\text{train}}} \log \left(\frac{\exp(h^L_{p,y_p})}{\sum_{k=1}^{K} \exp(h^L_{p,k})} \right) .$$

Graph clustering

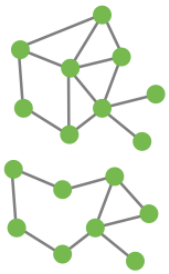
Data: $G_1, \dots G_n$ and labels on graphs.

Goal: Learn the Classification function $f : G \mapsto \{1, \dots, K\}$

Architecture:

$\mathcal{G}_1 = (\mathbf{X}_1, \mathbf{A}_1)$

$\mathcal{G}_2 = (\mathbf{X}_2, \mathbf{A}_2)$



$$\text{GNN}\left(\begin{bmatrix} \mathbf{A}_1 & \\ & \mathbf{A}_2 \end{bmatrix}, \begin{bmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \end{bmatrix}\right) = \begin{bmatrix} \mathbf{X}'_1 \\ \mathbf{X}'_2 \end{bmatrix}$$

and batches

Average over nodes in the same graph in order to have a layer at the graph level and use a classifier.

Loss: cross entropy.

Link Prediction

Data: $G = (V, E)$, V is incomplete.
Goal: Find edges that are likely to exist for a given set of non-observed edges...

Architecture: GCN layers with V as the set of edges... Last layer uses a "decoder" for dyads:

$$g(\text{Dist}(h_i^l, h_j^l)) \text{ or } h_i^{l\top} h_j^l$$

Loss: Cross entropy computed on a set of trainable DYADS (usually half of edges and half of non edges).

Remark: Autoencoder directly derived from link prediction task by using h_i^l as the embedding.

GVAE

Data: $G = (V, E)$.

Goal: Find an embedding of nodes in a small dimension (Euclidean) space as a conditional distribution.

Architecture: GCN layers to embed the nodes in the parameters of a Gaussian distribution, simulation under the distribution and a last decoder layer to predict edges.

$$(X_i)_i \to (m_i, s_i)_i \to (Z_i = m_i + s_i \cdot \mathcal{N}(0,1))_i \to (Z_i^\top Z_j)_{ij}$$

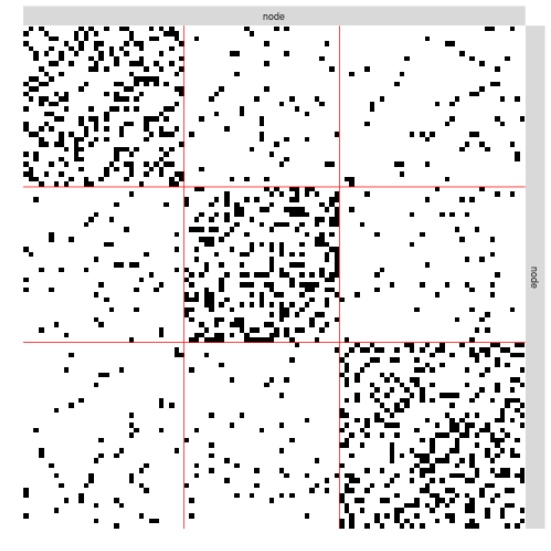
Loss: Cross entropy with a KL on the set of trainable DYADS:

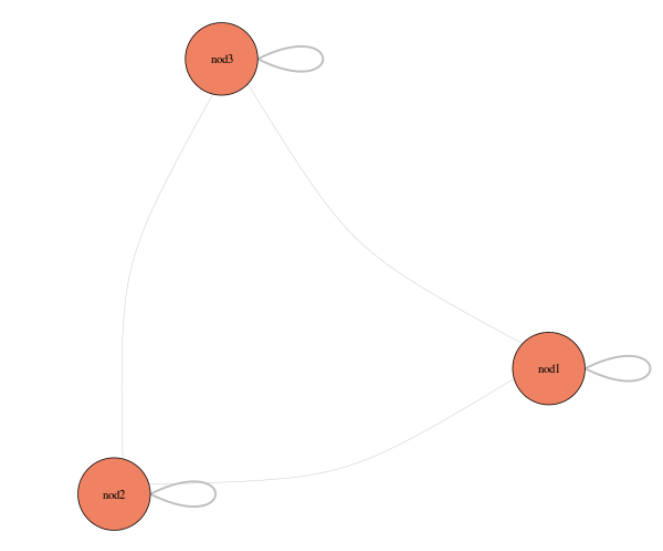
$$\mathbb{E}_{q(Z|X,A)}\big(\log p(A_{\text{train}}|Z)\big) - KL(q(Z|X,A)||p(Z))$$

where $p(Z)$ is a prior distribution chosen as $\mathcal{N}(0,1)$.

Some Examples

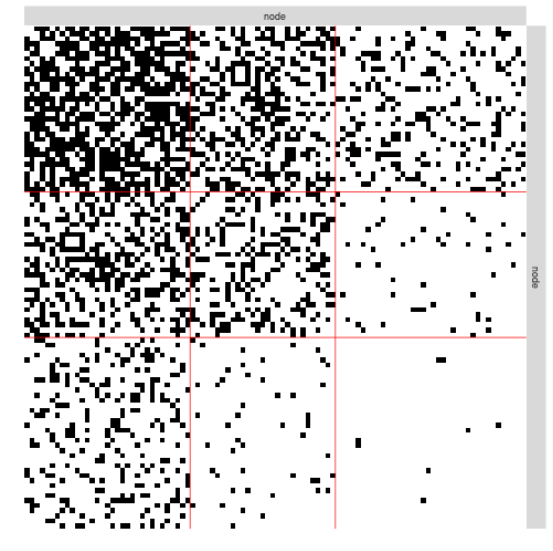
Communities

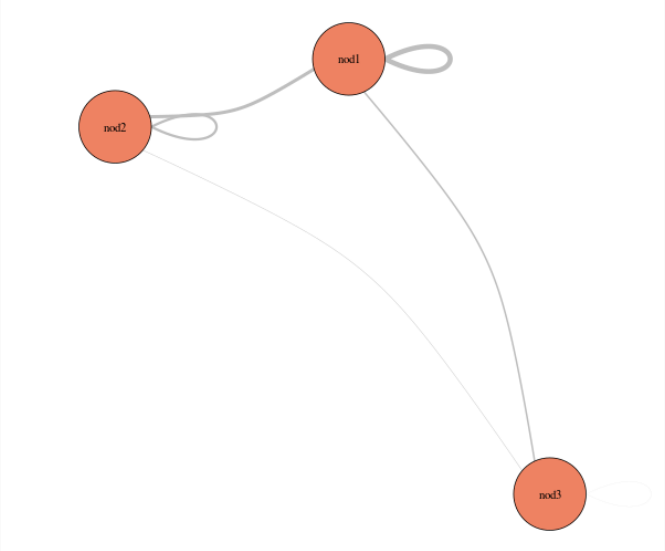




Some Examples

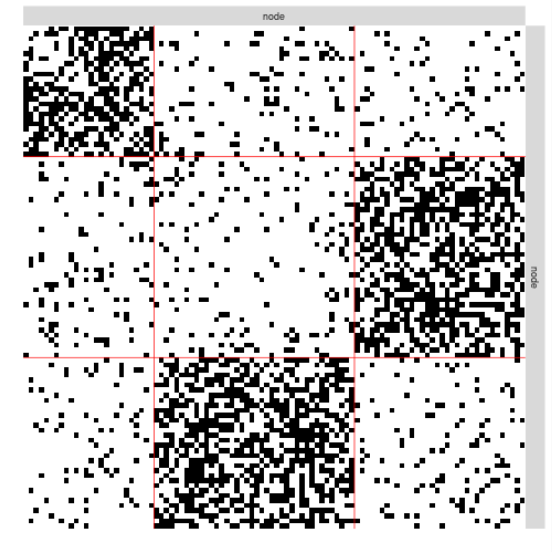
Nestedness

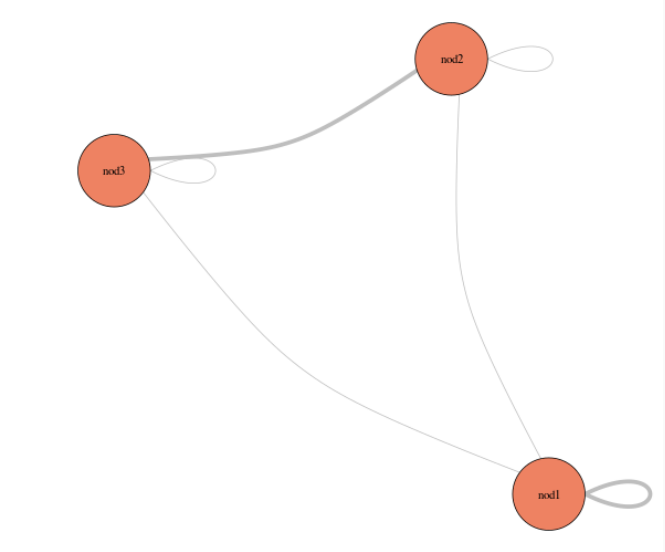




Some Examples

Community and antagonism





Many resources online

- introduction to GNN https://distill.pub/2021/gnn-intro/,
- convolution on graphs https://distill.pub/2021/understanding-gnns/,
- google colabs for pytorch geometric https://pytorch-geometric.readthedocs.io/en/latest/get_started/colabs.html.

Bonald, T. and De Lara, N. (2023).
A consistent diffusion-based algorithm for semi-supervised graph learning.
In International Conference on Complex Networks and Their Applications, pages 272–282. Springer.

Casanova, P., Lio, A. R. P., and Bengio, Y. (2018).
Graph attention networks.
ICLR. Petar Velickovic Guillem Cucurull Arantxa Casanova Adriana Romero Pietro Liò and Yoshua Bengio.

Defferrard, M., Bresson, X., and Vandergheynst, P. (2016).
Convolutional neural networks on graphs with fast localized spectral filtering.
Advances in neural information processing systems, 29.

Grover, A. and Leskovec, J. (2016).
node2vec: Scalable feature learning for networks.
In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, pages 855–864.

Hamilton, W., Ying, Z., and Leskovec, J. (2017).

Inductive representation learning on large graphs.
Advances in neural information processing systems, 30.


Kipf, T. N. and Welling, M. (2016).
Semi-supervised classification with graph convolutional networks.
arXiv preprint arXiv:1609.02907.


Matias, C. and Robin, S. (2014).
Modeling heterogeneity in random graphs through latent space models: a selective review.
ESAIM: Proceedings and Surveys, 47:55–74.


Nowicki, K. and Snijders, T. A. B. (2001).
Estimation and prediction for stochastic blockstructures.
Journal of the American Statistical Association, 96(455):1077–1087.


Peter D Hoff, A. E. R. and Handcock, M. S. (2002).
Latent space approaches to social network analysis.
Journal of the American Statistical Association, 97(460):1090–1098.


Rubin-Delanchy, P., Cape, J., Tang, M., and Priebe, C. E. (2022).
A statistical interpretation of spectral embedding: the generalised random dot product graph.

Journal of the Royal Statistical Society Series B: Statistical Methodology, 84(4):1446–1473.

Von Luxburg, U. (2007).
A tutorial on spectral clustering.
Statistics and computing, 17:395–416.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2018).
How powerful are graph neural networks?
arXiv preprint arXiv:1810.00826.