Spectral partitioning:

Given $L = V \Lambda V^T$

Find a clustering into k groups:

① Take:

$$V_k = \begin{bmatrix} \vec{v}_1 & \cdots & \vec{v}_k \end{bmatrix}_{N \times k} \xrightarrow{} \vec{w}_i \text{ (row)}$$

$\vec{w}_i$ $k \times 1$

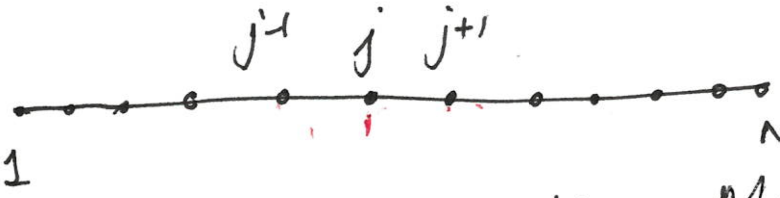
$\parallel$

② ~~Take the~~ rows of $V_k(i,:)$ as descriptors of each node.

③ Carry out k-means on the vectors $\{\vec{w}_i\}_{i=1}^N$ $\vec{w}_i \in R^k$

Connection of graphs with random walks.

Consider discretization of a 1D domain

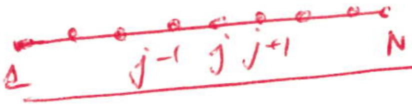


Discrete - time Markov chain

Defined in terms of: $(\vec{p}_t)_{N\times 1}$ $\equiv$ each coordinate contains the probability of the random walker at node i and time t.

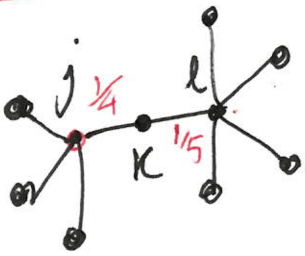
$$\vec{1}^T \vec{p}_t = 1. \quad \text{(Normalized)}$$

Random walk on:



$$p_{t+1}^{(j)} = \frac{1}{2}\left(p_t^{(j-1)} + p_t^{(j+1)}\right)$$

This applies generally to any graph:



$$p_{t+1}^{(k)} = \frac{1}{5} p_t^{(l)} + \frac{1}{4} p_t^{(j)}$$

For a given graph with adjacency matrix A we have a Discrete-time Markov chain:

$$\vec{p}_{t+1} = \underbrace{(AD^{-1})}_{\overset{\shortparallel}{M}} \vec{p}_t = M \vec{p}_t$$

Transition matrix

Solution: $\vec{p}_t = M^t \vec{p}_0$

Associated continuous-time process:

$$\frac{d\vec{p}(t)}{dt} = -\underbrace{(I - AD^{-1})}_{\overset{\shortparallel}{L_{RW}}} \vec{p}(t)$$

$$L_{RW} D = L = D - A$$

$$L_{RW} = L D^{-1}$$

And the isospectral (with L_{RW})

Symmetrized normalized Laplacian:

$$\mathcal{L} = D^{-1/2} L_{RW} D^{1/2} = D^{-1/2} L D^{-1/2}$$

$\underline{\mathcal{L} = \mathcal{L}^T}$ ✓

See 'Spectral clustering' section above.

⇒ Similar algorithms based on eigenvectors of $\mathcal{L}$.

(Ng & Jordan)

Note: They normalize by row too!

---

(2) Alternative to spectral clustering:

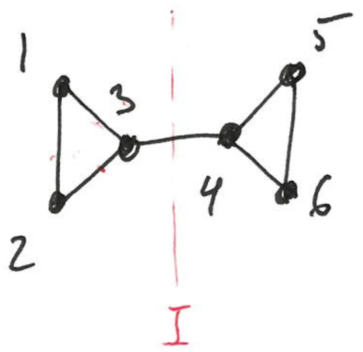
by

<u>Modularity</u>. (Newman)

Find a partition of the graph that will have maximally block diagonal structure in the adjacency matrix, and has more 'blocks' than expected at random.

Idea behind modularity: Count edges within blocks and between blocks

Example:



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

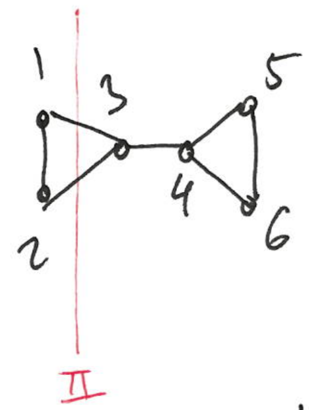
$$H_I = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}_{N \times K}$$

Into $\underline{\underline{c}}$ groups

$\underline{c=2}$

$$\frac{1}{2}\left(H_I^T A H_I\right)_{c \times c} = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} A \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} =$$

$$= \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 2 & 0 \\ 2 & 1 \\ 1 & 2 \\ 0 & 2 \\ 0 & 2 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 6 & 1 \\ 1 & 6 \end{bmatrix} = \begin{bmatrix} 3 & \frac{1}{2} \\ \frac{1}{2} & 3 \end{bmatrix}$$



Same A

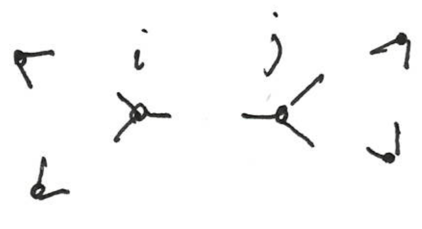
$$H_{II} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

$$\frac{1}{2}\left(H_{II}^{T} A H_{II}\right) = \begin{bmatrix} 1 & 1 \\ 1 & 4 \end{bmatrix}$$

$$\boxed{\underset{H}{\text{Maximise}} \quad \mathrm{Tr}\left[\frac{1}{2}\left(H^{T} A H\right)\right]}$$

Second ingredient : Null model.

<u>Configuration model</u>.



$$\vec{d} = \begin{pmatrix} d_1 \\ \vdots \\ d_N \end{pmatrix} \qquad 2E = \sum_{i=1}^{N} d_i$$

$$R_{ij} = \frac{d_i d_j}{2E}$$

Discount the edges that are expected to be present at random.

$$R = \frac{1}{2E} \vec{d}\,\vec{d}^T$$

(Expected edges if we rewire at random)

where $\vec{d} = A\vec{1}$

Construct the modularity matrix: Z

$$\frac{1}{2E} \mathrm{Tr}\left[H^T \underbrace{\left[A - \frac{1}{2E} \vec{d}\,\vec{d}^T \right]}_{Z} H \right] = Q$$

and try to make it as modular as possible:

$$\boxed{\begin{array}{c}\text{Modularity optimisation:} \\ \max\limits_{H} Q\end{array}}$$

How is Q optimised:

① Similarly to spectral clustering, it can be shown that we can use the leading eigenvector of Z to maximise Q.

Relaxation to $\vec{s} \in \mathbb{R}^N$, etc

Then we can effect bipartitions in a recurrent manner until Q does not increase.

Stopping criterion: $\Delta Q < 0$ in the iteration.

② In reality modularity is optimised through a greedy agglomerative algorithm that performs better than the spectral relaxation ~~bon~~ the eigenvectors of Z.

This method is called <u>Louvain optimisation</u> and has become an industry standard.

It was proposed by Blondel et al in 2008.