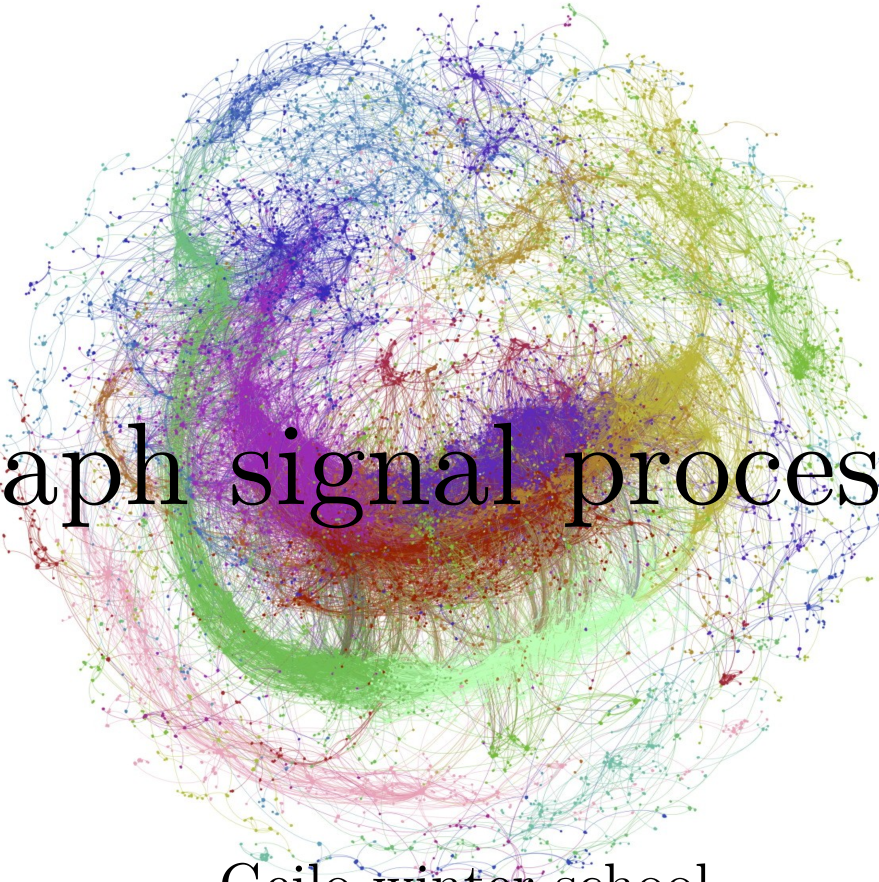


Graph signal processing

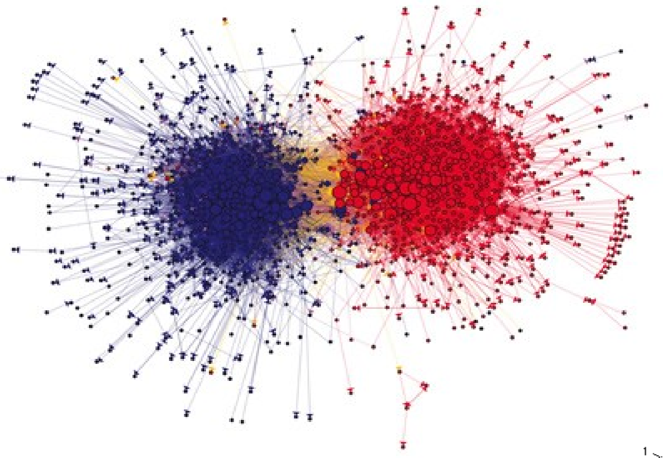
Geilo winter school
January 2024

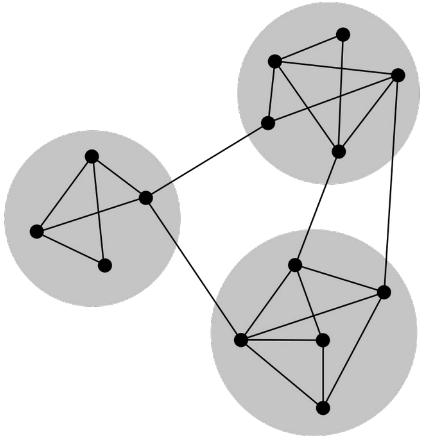
Benjamin Ricaud, The Arctic University of Norway, Tromsø

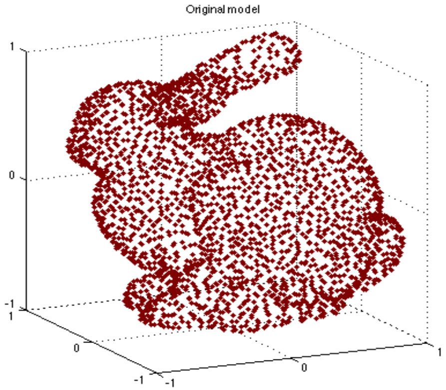
Some references available online:

- F. Chung, Lectures on Spectral Graph Theory:
  https://mathweb.ucsd.edu/~fan/research/cbms.pdf
- U. von Luxburg, A Tutorial on Spectral Clustering:
  https://www.cs.cmu.edu/~aarti/Class/10701/readings/Luxburg06_TR.pdf
- D. A. Spielman Spectral and Algebraic Graph Theory:
  http://cs-www.cs.yale.edu/homes/spielman/sagt/sagt.pdf
- C. O. Aguilar, An Introduction to Algebraic Graph Theory:
  https://www.geneseo.edu/~aguilar/public/notes/Graph-Theory-HTML/index.html
- Label propagation in scikit-learn:
  https://scikit-learn.org/stable/modules/semi_supervised.html#label-propagation

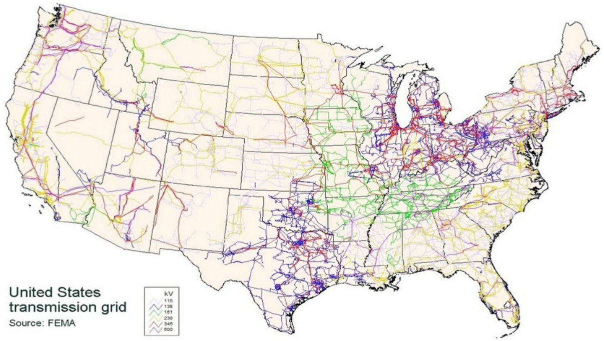
Introduction





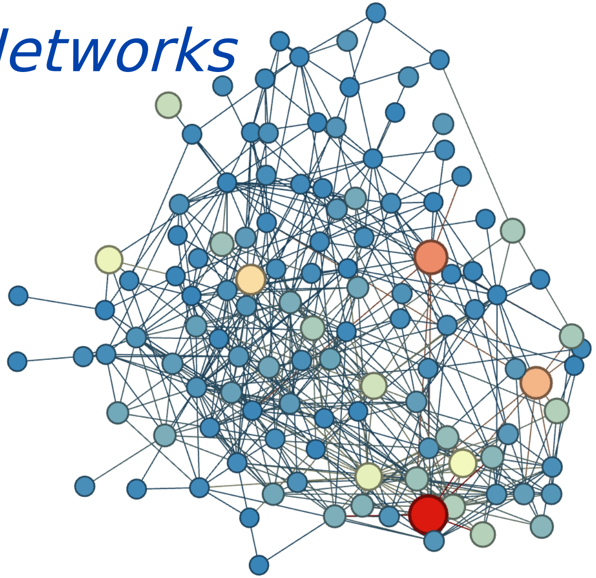


Graph / Networks

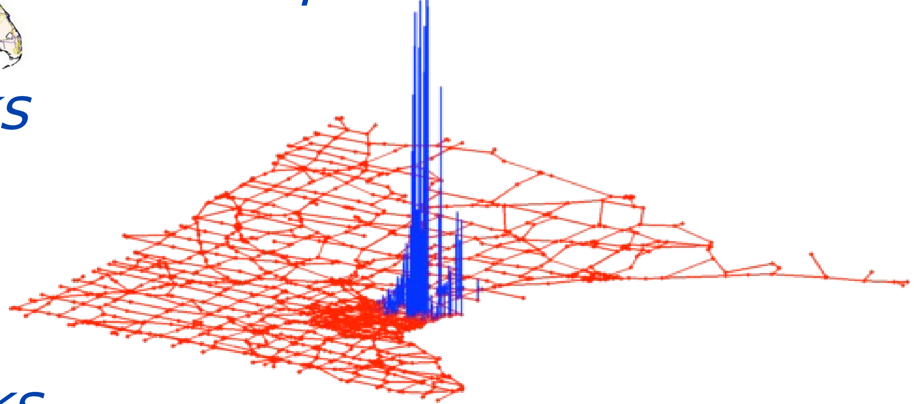


Energy Networks

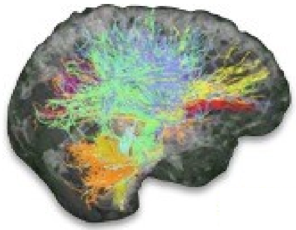
Social/web Networks



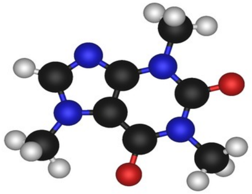
Transportation Networks



Biological Networks

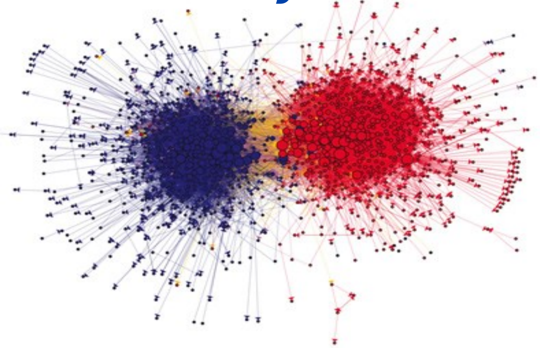


Chemistry

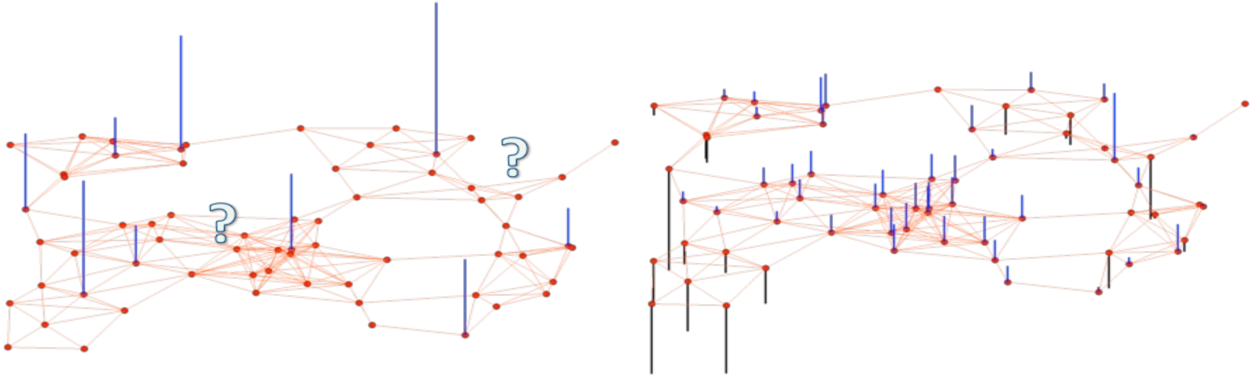


Typical applications

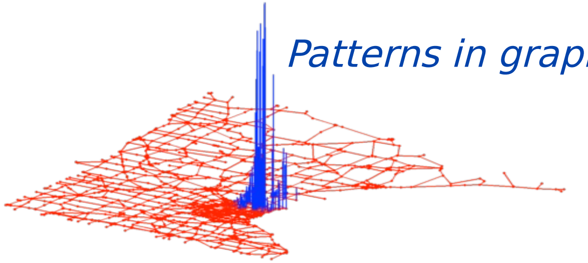
Graph only
(Community detection)



Graph + values

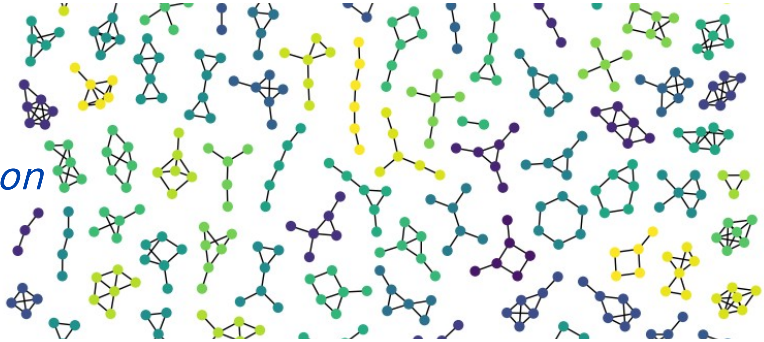


Pattern detection with Graph Neural Nets



Patterns in graphs

Graph classification



Graph signal processing

Graph + values on the nodes

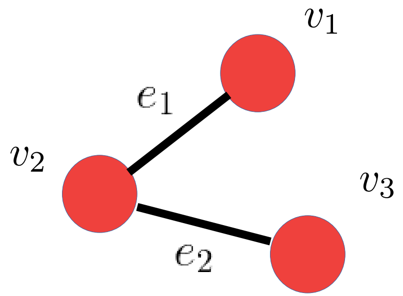
Graph definition

Vertices or nodes

$V = \{v_1, \ldots, v_N\}$

Edges or links

$E = \{e_1, \ldots, e_M\}$



Adjacency matrix

$$A = \begin{array}{cc} & \begin{array}{ccc} v_1 & v_2 & v_3 \end{array} \\ & \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{array}{c} v_1 \\ v_2 \\ v_3 \end{array} \end{array}$$

$$\mathbf{A}(i,j) = \begin{cases} +1 & \text{if there is an edge } (v_i, v_j) \text{ or } (v_j, v_i) \in E \\ 0 & \text{otherwise} \end{cases}$$

All the information on the graph is contained in the adjacency matrix

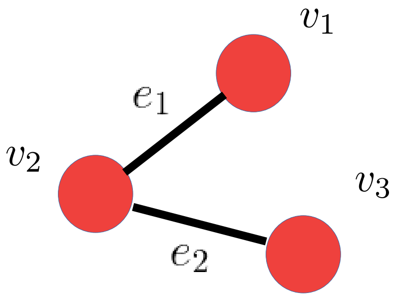
Graph definition

Vertices or nodes

$$V = \{v_1, \dots, v_N\}$$

Edges or links

$$E = \{e_1, \dots, e_M\}$$



Weight Matrix:

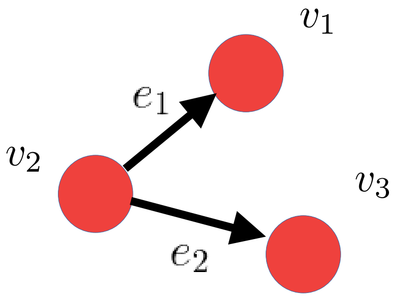
$$W = \begin{pmatrix} 0 & w_{12} & 0 \\ w_{21} & 0 & w_{23} \\ 0 & w_{32} & 0 \end{pmatrix}$$

W*(i,j)* is the weight ("strength") of the edge between *i,j* (if any).
W symmetric with positive entries

Directed Graph

Weight Matrix:

$$W = \begin{pmatrix} 0 & 0 & 0 \\ w_{21} & 0 & w_{23} \\ 0 & 0 & 0 \end{pmatrix}$$



There is a connection from v_2 to v_1 but not from v_1 to v_2.
(example: hyperlinks in webpages)

W with positive entries but not symmetric. $W \neq W^T$

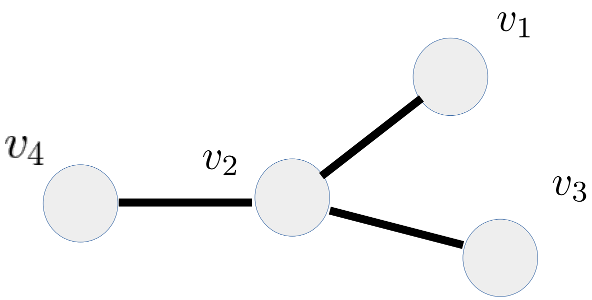
Another important matrix, the degree matrix D

- Degree matrix D: diagonal matrix with node degree on the diagonal,

$$D_{ii} = \sum_j w_{ij} = d_i$$

Degree of node i

Small quiz



Adjacency matrix ?
Degree matrix ?

Part 1:

Some properties of A and W and why we need them

Matrix properties

- Natural question:
- What are the eigenvalues / eigenvectors, the "spectrum" ?

"Spectral graph theory"

Remark: Any symmetric matrix has a set of orthogonal eigenvectors, that is why undirected graphs are more studied.

Methods where they play a role

- PageRank
- Label propagation
- Spectral clustering
- Graph signal processing
- Graph neural networks

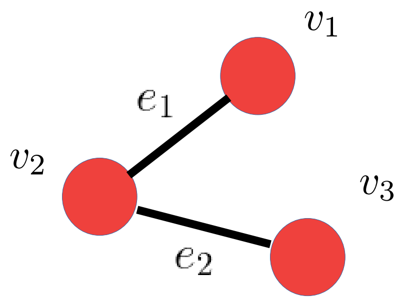
Most of these methods uses the Laplacian matrix instead of the weight matrix. This is for several reasons and we will see why soon.

Transition matrix

- Adjacency matrix with normalized columns (or rows)

$$T_{ij} = \frac{a_{ij}}{\sum_i a_{ij}} = \frac{a_{ij}}{d_j}$$

Degree of node j



$$A = \begin{array}{c} \begin{array}{ccc} v_1 & v_2 & v_3 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \end{array} \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \end{array}$$

Probability to jump from node j to node i

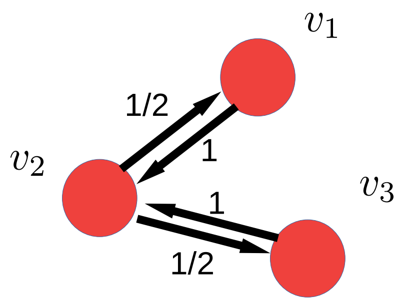
Transition matrix

$$T = \begin{array}{c} \begin{array}{ccc} v_1 & v_2 & v_3 \end{array} \\ \begin{pmatrix} 0 & 1/2 & 0 \\ 1 & 0 & 1 \\ 0 & 1/2 & 0 \end{pmatrix} \end{array} \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \end{array}$$

- Not symmetric

$$\begin{array}{c|ccc|c} & v_1 & v_2 & v_3 & \\ & & & & \\ \end{array}$$

$$T = \begin{array}{c} \\ \end{array}\begin{pmatrix} 0 & 1/2 & 0 \\ 1 & 0 & 1 \\ 0 & 1/2 & 0 \end{pmatrix}\begin{array}{c} v_1 \\ v_2 \\ v_3 \end{array}$$



- Degree matrix D: diagonal matrix with node degree on the diagonal,

$$D_{ii} = \sum_j w_{ij} = d_i$$

Degree of node i

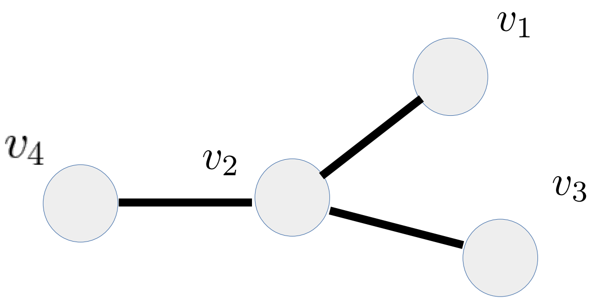
Transition matrix or **random walk matrix**

$$T = W D^{-1}$$

$$T_{ij} = [W D^{-1}]_{ij} = \frac{w_{ij}}{d_j}$$

Probability to jump from j to i

Small quiz



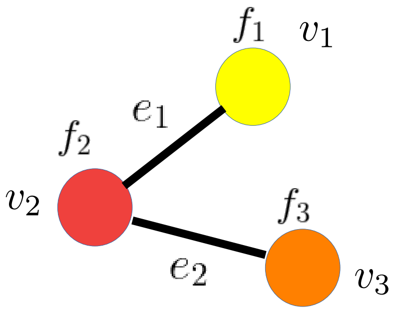
Transition matrix ?

Graph + values in the nodes

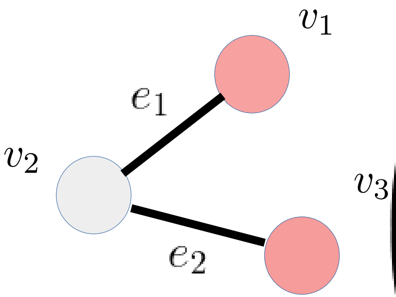
Encoding values on a graph

- With a vector

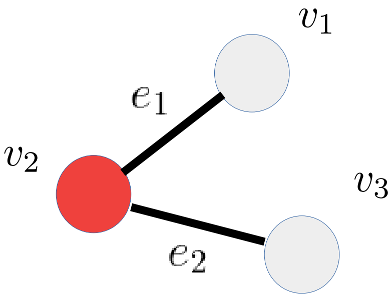
$$\begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix}$$



Propagation on a graph



$$\begin{array}{ccccc} & & \text{T} & & \text{Y} \\ \begin{pmatrix} 1/2 \\ 0 \\ 1/2 \end{pmatrix} & = & \begin{pmatrix} 0 & 1/2 & 0 \\ 1 & 0 & 1 \\ 0 & 1/2 & 0 \end{pmatrix} & & \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \end{array}$$



Split equally between node 1 and 3

Transition matrix

Value 1 on node 2

- Applying T to the vector of node values propagate the values over the graph
- Probabilistic interpretation of T^nY: probability to reach an node (starting from v_2) after exactly n steps

PageRank

Famous Google PageRank, from Larry Page, Sergey Brin (1998)

Idea:

- Explore webpages by following the hyperlinks randomly
- When a page is visited, increase its score
- To avoid being stuck on pages without links, add a probability to randomly jump to any other page.

This is equivalent to a particular random walk on the graph

Iterate $Y_{t+1}=MY_t$

with Y_0 random vector

$$M = \underbrace{(1-p)}_{\text{Tuning parameter}}\underbrace{T}_{\text{Transition matrix}} + p\underbrace{B}_{\text{Random jump anywhere matrix}}$$

$$B = \frac{1}{n} \cdot \begin{bmatrix} 1 & 1 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{bmatrix}$$

The math behind

The method consist in applying a matrix many times to an initial vector.
$Y_n = M^n Y_0$
Provided some conditions on M, this converge when $n \to \infty$ to a unique solution.
Conditions:
matrix with non-negative entries
Irreducible matrix: any node in the graph can be reached from any other node.
Perron-Frobenius theorem:

- S is an eigenvector of M associated to the largest eigenvalue $\lambda > 0$,
- S is the unique eigenvector with eigenvalue λ,
- Any other eigenvalue $|\lambda_i| < \lambda$

In our case the largest eigenvalue is 1, the others are smaller but positive.
$\rightarrow$ we can use the power method to find S. Because for any eigenvector U_i:
$M^n U_i = \lambda_i^n U_i$ and $\lambda_i < 1$ except for λ
So will converge to S as n increases

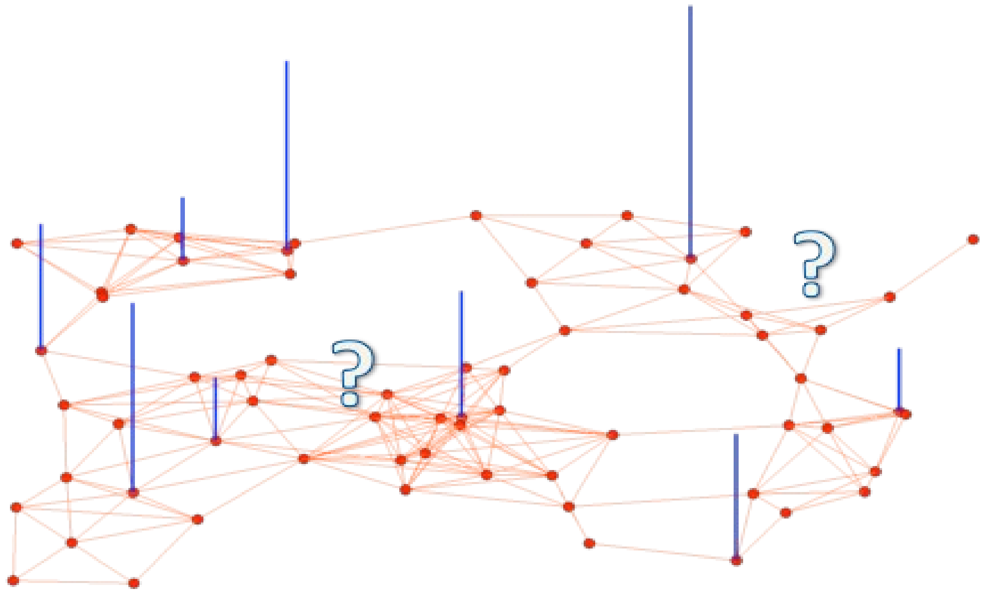
$$Y = \sum_{j=0}^{N-1} \lambda_i U_i$$

If max eigenvalue *m* is above 1: divide by the norm after each "diffusion": $||M^n U||$ to prevent an exponential increase.

Power iteration method

```python
import numpy as np

def power_iteration(A, num_iterations: int):
    # Ideally choose a random vector
    # To decrease the chance that our vector
    # Is orthogonal to the eigenvector
    b_k = np.random.rand(A.shape[1])

    for _ in range(num_iterations):
        # calculate the matrix-by-vector product Ab
        b_k1 = np.dot(A, b_k)

        # calculate the norm
        b_k1_norm = np.linalg.norm(b_k1)

        # re normalize the vector
        b_k = b_k1 / b_k1_norm

    return b_k
```

Propagation of labels



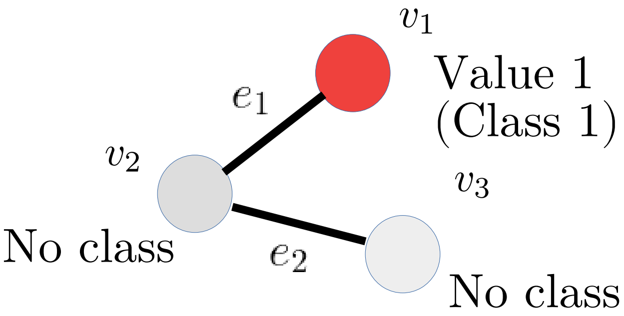
Idea: connected nodes share similar properties (similar label, value or vector of values)

Propagation of labels

node 1 has value 1

Vector associated with a label:

$$Y = \begin{array}{cc} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} & \text{nodes} \end{array}$$



Initial values Y_0
Propagate: $Y_{t+1} = TY_t$

Ok but we need to keep the labelled nodes at their initial value!

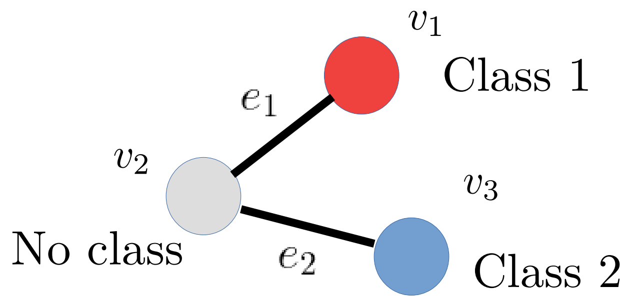
$\rightarrow$ Propagate and clamp the labelled nodes: $Y_{t+1} = TY_t + Y_0$
Under some conditions on T it converges

From: Learning from Labeled and Unlabeled Data with Label Propagation, Xiaojin Zhu and Zoubin Ghahramani, 2002.

Propagation of labels

Several classes: Y is a matrix, each column is a label

$$\begin{array}{ccc} & \text{Classes} & \\ Y = & \begin{pmatrix}1 & 0\\ 0 & 0\\ 0 & 1\end{pmatrix} & \text{nodes}\end{array}$$



Probability of node 3 to be of class 2

All labels are diffused at the same time

Alternative propagation

Zhou, D., Bousquet, O., Lal, T., Weston, J., & Schölkopf, B. (2003). Learning with local and global consistency. Advances in neural information processing systems, 16.

1. Form the affinity matrix W defined by $W_{ij} = \exp(-\|x_i - x_j\|^2/2\sigma^2)$ if $i \neq j$ and $W_{ii} = 0$.
2. Construct the matrix $S = D^{-1/2}WD^{-1/2}$ in which D is a diagonal matrix with its (i, i)-element equal to the sum of the i-th row of W.
3. Iterate $F(t+1) = \alpha SF(t) + (1-\alpha)Y$ until convergence, where α is a parameter in $(0, 1)$.
4. Let F^* denote the limit of the sequence $\{F(t)\}$. Label each point x_i as a label $y_i = \arg\max_{j \leq c} F^*_{ij}$.

$D^{-1/2}WD^{-1/2}$ Symmetrized transition matrix

Break

- Let us make some diffusion / propagation on a graph.
- Take you favorite graph, define a signal
- Choose an algorithm and apply the matrix iteratively
- Visualize the evolution. For simple visualization you may use a path graph.