Statistical Inference over Graphs

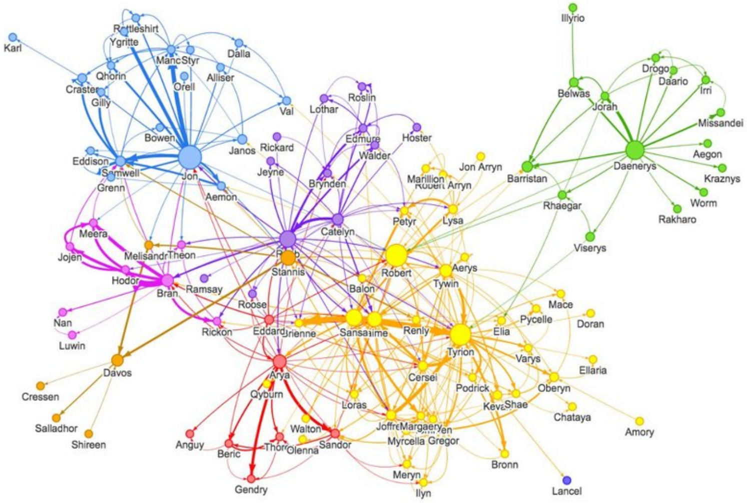
Philippe Ciblat

Outline

Part 1 : Introduction to Graph Theory

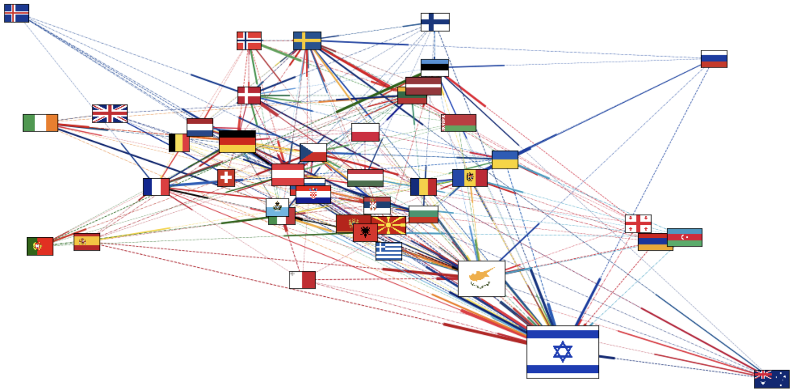
Where do you find graphs ?

Social Networks



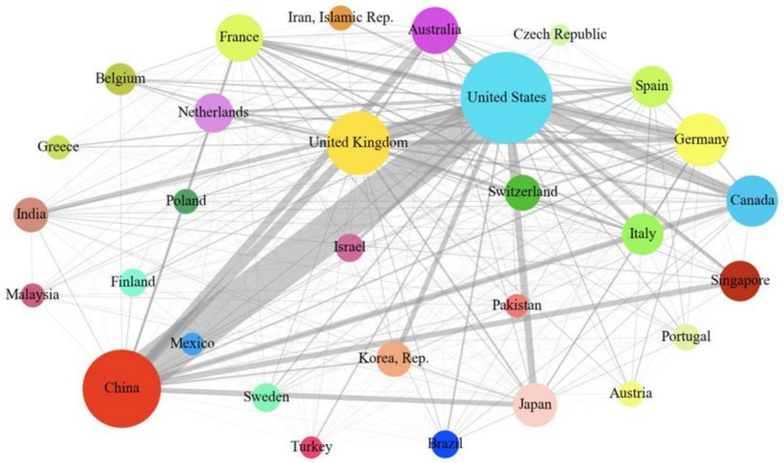
Where do you find graphs ?

Vote Networks



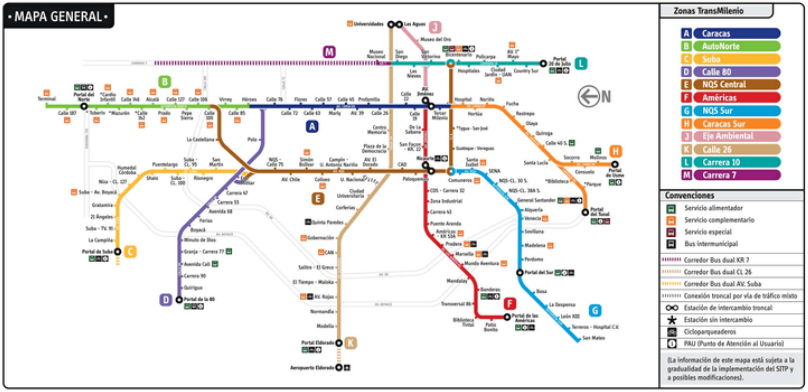
Where do you find graphs ?

Papers' database



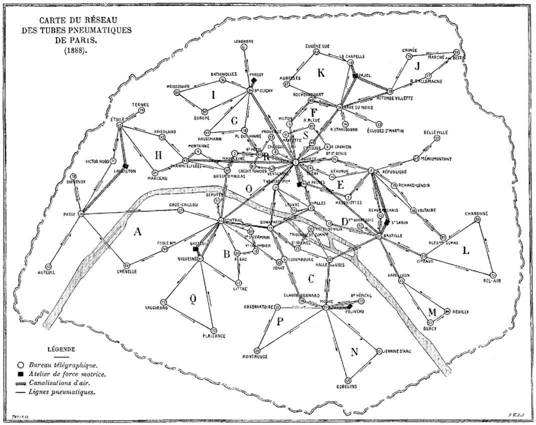
Where do you find graphs ?

Public Transportation map



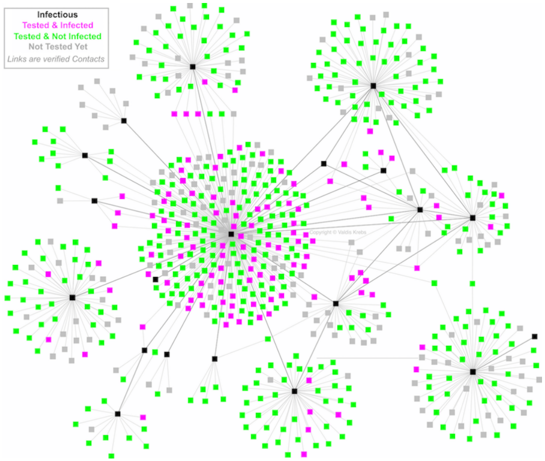
Where do you find graphs ?

Communication Networks



Where do you find graphs ?

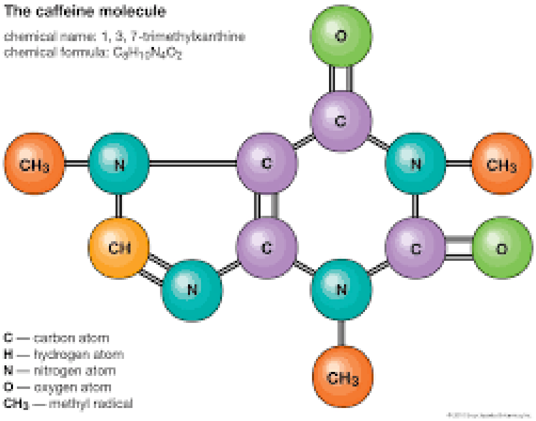
Human interaction for epidemic propagation analysis



source: V. Krebs, "Tracking and stopping the spread of a contagious disease"

Where do you find graphs ?

Molecule Networks



Graph characteristics

Two main types of graphs

- Non-attributed graphs
  - N nodes/vertices
  - Links/Edges between some nodes
  - Edges may be directed/non-directed and/or weighted/non-weighted
- Attributed graphs
  - Each node i has also a feature/value $\mathbf{x}_i \in \mathbb{R}^K$

Mathematical representations

Here, we consider non-directed and non-weighted graphs.

- Let i be a node and $\mathcal{N}_i$ be the set of its neighbors
- Node degree: $d_i = |\mathcal{N}_i|$ (number of neighbors)
- Degrees matrix: $\mathbf{D} = \mathrm{diag}(d_1, \cdots, d_N)$
- Adjacency matrix: $\mathbf{A}$

$$a_{ij} = \begin{cases} 1 & \text{if } i \text{ and } j \text{ are connected} \\ 0 & \text{otherwise} \end{cases}$$

  Be careful: $a_{ii} = 0$
- Laplacian matrix: $\mathbf{L} = \mathbf{D} - \mathbf{A}$

Some results

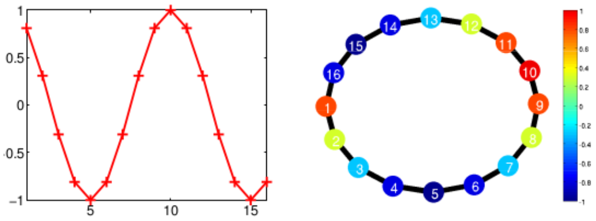
- $\mathbf{L}.\mathbf{1} = \mathbf{0}$
- The second smallest eigenvalue $\lambda_2 \neq 0$ iff graph is connected

Graph decomposition (1/3)

- Timeline is a graph: if periodic signal (of length N), it is even a ring graph
  - $y_n = \sum_k h_k s^k(x_n)$ with $s(x_n) = x_{n-1}$
  - $\mathbf{y} = [y_{N-1}, \cdots, y_0]^{\mathrm{T}}$. Then

$$\mathbf{y} = \mathbf{C}\mathbf{x} \text{ and } \mathbf{C} = \mathbf{F}\mathbf{M}\mathbf{F}^{\mathrm{H}}$$

  with $\mathbf{M} = \operatorname{diag}(m_0, \cdots, m_{N-1})$ and $m_\ell = \sum_{n=0}^{N-1} h_k e^{-2i\pi \ell k/N}$
  - Then $\mathbf{X} = \mathbf{F}^{\mathrm{H}}\mathbf{x}$ is the Fourier Transform of any vector $\mathbf{x}$.



source: N. Tremblay, "Networks and Signal: signal processing tools for networks analysis", PhD thesis, ENS Lyon, 2014

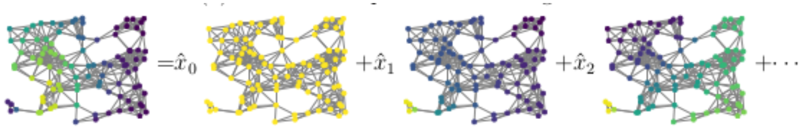
Graph decomposition (2/3)

- For any graph
  - $\mathbf{y} = \sum_k h_k s^k(\mathbf{x})$ with $s(\mathbf{x})_m = \sum_{\ell \in \mathcal{N}_m} s_{m,\ell} x_\ell$
  - $\mathbf{y} = \sum_k h_k \mathbf{S}^k \mathbf{x}$. If $\mathbf{S} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^{\mathrm{T}}$ Then

$$\mathbf{y} = \mathbf{V} \left(\sum_k h_k \boldsymbol{\Lambda}^k \right) \mathbf{V}^{\mathrm{T}} \mathbf{x}$$

  - By analogy
    - $m_\ell = \sum_k h_k \lambda_\ell^k$ with $\lambda_\ell = e^{-2i\pi\ell/N}$, $\mathbf{F} = \mathbf{V}$, and $\mathbf{S} = \mathrm{circ}([0, 1, 0, \cdots])$
    - Conversely, $\mathbf{X} = \mathbf{V}^{\mathrm{T}}\mathbf{x}$ is called Graph Fourier Transform
    - $\mathbf{v}_\ell$ (ℓ-th column of $\mathbf{V}$) is the ℓ-th Fourier Graph as $\mathbf{y} = m_\ell \mathbf{v}_\ell$ if $\mathbf{x} = \mathbf{v}_\ell$
    - And $Y_\ell = m_\ell X_\ell$.

Graph decomposition (3/3)



A. Barbe, "Diffusion-Wasserstein distances for attributed graphs", PhD thesis, ENS Lyon, Dec 2021

Application: Heat diffusion

Continuous-time Heat diffusion within a graph

- A time t, the temperature of the node ℓ is denoted by $x_\ell(t)$
- The update law comes from Heat diffusion equation

$$\frac{dx_\ell}{dt} = -\sum_{m\in\mathcal{N}_\ell}(x_\ell - x_m) \Leftrightarrow \frac{d\mathbf{x}}{dt} = -\mathbf{L}\mathbf{x}$$

- The solution is

$$\mathbf{x}(t) = e^{-t\mathbf{L}}\mathbf{x}(0)$$

- Let $\mathbf{L} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{\mathrm{T}}$ with $\lambda_1 = 0$ and $\mathbf{v}_1 = \mathbf{1}/\sqrt{N}$. Then

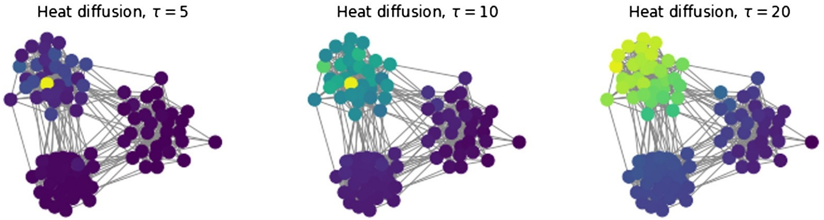
$$e^{-t\mathbf{L}} = \mathbf{V}e^{-t\mathbf{\Lambda}}\mathbf{V}^{\mathrm{T}} \overset{t\to\infty}{\to} \mathbf{v}_1\mathbf{v}_1^{\mathrm{T}} = \frac{1}{N}\mathbf{1}\mathbf{1}^{\mathrm{T}}$$

Therefore

$$\lim_{t\to\infty}\mathbf{x}(t) = \overline{x}\mathbf{1}$$

with $\overline{x}$ the average of initial temperatures.
The speed of convergence depends on the graph through $\{\lambda_\ell\}_{\ell=2,\cdots,N}$

Application: Numerical illustrations



B. Ricaud, P. Borgnat, N. Tremblay, P. Gonçalves, P Vandergheynst, "Fourier could be a data scientist: from Graph Fourier transform to signal processing on graphs", Comptes-rendus de Physique de l'Académie des Sciences, Aug. 2019

Application: Consensus algorithm (1/2)

We start with an initial value $\mathbf{x}(0)$.

- At time t, one node wakes up (let's say ℓ)
  - Algorithm 1 (pairwise): ℓ selects $\ell' \in \mathcal{N}_\ell$ and

$$x_\ell(t+1) = x_{\ell'}(t+1) = \frac{x_\ell(t) + x_{\ell'}(t)}{2}$$

  - Algorithm 2 (broadcast): ℓ broadcasts its value at any neighbor who updates

$$x_{\ell'}(t+1) = \frac{x_\ell(t) + x_{\ell'}(t)}{2}, \ \ell' \in \mathcal{N}_\ell$$

Finally

$$\mathbf{x}(t) = \prod_{k=1}^{t} \mathbf{W}_k \mathbf{x}(0)$$

Results

- Algorithm 1: convergence to $\overline{x}$ (as $\lim_{t\to\infty} \prod_{k=1}^{t} \mathbf{W}_k = \frac{1}{N}\mathbf{1}\mathbf{1}^{\mathrm{T}}$)
- Algorithm 2: convergence to $\underline{x} = \mathbf{v}_\infty^{\mathrm{T}}\mathbf{x}(0)$ (as $\lim_{t\to\infty} \prod_{k=1}^{t} \mathbf{W}_k = \mathbf{1}\mathbf{v}_\infty^{\mathrm{T}}$)

Application: Consensus algorithm (2/2)

- Row-stochastic matrix: non-negative matrix and row sums to 1

$$\mathbf{W}\mathbf{1} = \mathbf{1} \Rightarrow \lim_{k\to\infty} \mathbf{W}^k = \mathbf{1}\mathbf{v}^{\mathrm{T}} (\text{mild conditions})$$

  with $\mathbf{v}^{\mathrm{T}}\mathbf{1} = 1$

- Column-stochastic matrix: non-negative matrix and column sums to 1

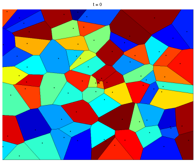
$$\mathbf{1}^{\mathrm{T}}\mathbf{W} = \mathbf{1}^{\mathrm{T}} \Rightarrow \lim_{k\to\infty} \mathbf{W}^k = \mathbf{v}\mathbf{1}^{\mathrm{T}} (\text{mild conditions})$$

Extension exists for a sequence of $\mathbf{W}_k$

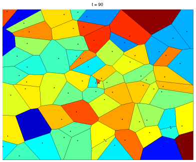
Remark

- For Algorithm 1: both row and column-stochastic matrix
- For Algorithm 2: only row-stochastic matrix

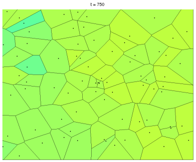
Application: Numerical illustrations



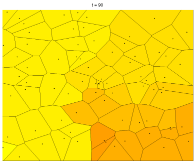
Initial Graph



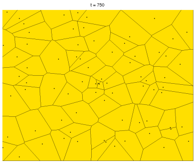
Pairwise ($t = 10$)



Pairwise ($t = 75$)

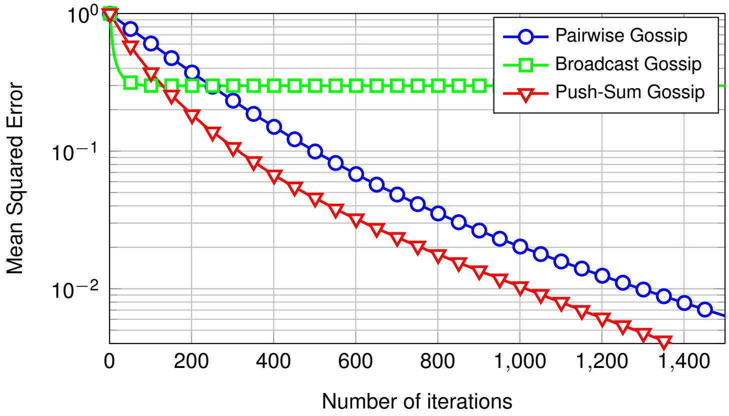


Broadcast ($t = 10$)



Broadcast ($t = 75$)

Application: Numerical illustrations

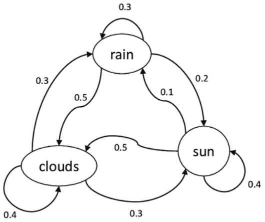


Link with Markov chain

Finite-state Markov Chain: state $s \in \mathcal{S} = \{s^1, \cdots, s^N\}$

$$\Pr\left(s_{t+1} = s^\ell | s_t = s^k\right) = T_{k,\ell} \geq 0$$

with $\sum_\ell T_{k,\ell} = 1$, so $\mathbf{T}$ is row-stochastic matrix



$$\begin{array}{c|ccc} & \text{clouds} & \text{rain} & \text{sun} \\ \hline \text{clouds} & 0.4 & 0.3 & 0.3 \\ \text{rain} & 0.5 & 0.3 & 0.2 \\ \text{sun} & 0.5 & 0.1 & 0.4 \end{array}$$

Analyzing Markov chain is equivalent to analyzing Graph

Stationary distribution: μ s.t. $\mu = \mu\mathbf{T}$

source: H. Seyr and M. Muskulus, "Decision Support Models for Operations and Maintenance for Offshore Wind Farms: A Review", Applied Sciences, April 2019

What do we want to do?

- Analyzing some math operators on Graph: see Graph Fourier Transform
- Analyzing convergence on Graph: see Markov chain, Gossip algorithms

  but in previous cases, graph is a tool, not the signal of interest

- In the remainder of the presentation, **Graph is the signal**
  - classification/clustering
  - node inference
  - link prediction (not done here)

Two types of graphs

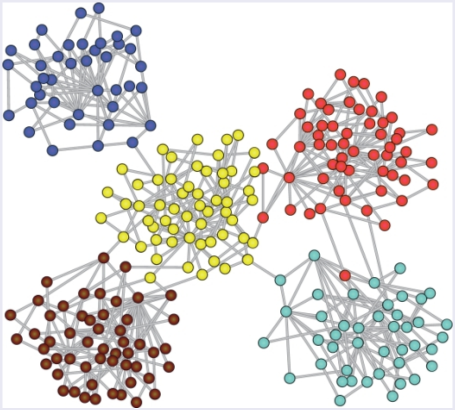
- Non-attributed (only links)
- Attributed (links + values)

Analysis is usually different (except if embedding procedure)

Part 2: Non-attributed graphs

2.1: Community detection

Example



Reminder: clustering based only on connection properties (no-valued nodes)

Modularity principle (1/2)

Clustering = Graph partition: $\Pi = \{\mathcal{P}_\ell\}_\ell$ s.t. $\mathcal{G} = \cup_\ell \mathcal{P}_\ell$ and $\cap_\ell \mathcal{P}_\ell = \emptyset$

$$\Pi^\star = \arg\max_{\Pi} Q(\Pi)$$

with Q the so-called modularity function

Assume random network. Let k and ℓ be two nodes with degrees d_k and d_ℓ. Let e be the number of edges
What is the probability for these two nodes to be connected ?

We have $2e$ stubs (stub: one link going from one node). Each stub of node k has a probability $d_\ell/(2e-1)$ to be connected to a stub of node ℓ. Therefore

$$\Pr\left(k \text{ connected to } \ell\right) = \frac{d_k d_\ell}{2e-1}$$

$$Q = \frac{1}{2e}\sum_{k,\ell}\left(a_{k\ell} - \frac{d_k d_\ell}{2e}\right)\delta_{\mathcal{P}_k,\mathcal{P}_\ell}$$

Modularity principle (2/2)

$$Q = \sum_{k,\ell} \frac{a_{k\ell}}{2e} \delta_{\mathcal{P}_k,\mathcal{P}_\ell} - \sum_{k,\ell} \frac{d_k}{2e}\frac{d_\ell}{2e} \delta_{\mathcal{P}_k,\mathcal{P}_\ell}$$

Let f_{ij} fraction of edges in the graph connecting nodes from cluster i to cluster j (factor 2 since undirected edge counts twice)

$$f_{ij} = \sum_{k,\ell} \frac{a_{k\ell}}{2e} \delta_{k\in\mathcal{P}_i} \delta_{\ell\in\mathcal{P}_j}$$

Let $c_i = \sum_j f_{ij}$ be the fraction of edges connecting nodes to cluster i

$$Q = \sum_i f_{ii} - \sum_{k,\ell} \frac{d_k}{2e}\frac{d_\ell}{2e} \delta_{\mathcal{P}_k,\mathcal{P}_\ell} = \sum_i (f_{ii} - c_i^2)$$

If graph is random, $f_{k\ell} \approx c_k c_\ell$

Complexity issue

- Solution 1: decentralized algorithm
- Solution 2: embedding, then clustering algorithm (like k-means)

2.2: Coalition game

Non-centralized case

- each node makes its decision by itself by observing its neighborhood
- then we iterate synchronously or asynchronously.

Mathematical Tool: Coalition Game Theory

Within a coalition structure (actually graph partition), we associate these quantities to each coalition/cluster $\mathcal{P}_k$:

- **Revenue/reward/utility** $u(\mathcal{P}_k) \geq 0$ quantifies the worth of the coalition, with $u(\emptyset) = 0$.
- **Cost** $c(\mathcal{P}_k) \geq 0$ quantifies the cost of cooperation.
- **Value** $v(\mathcal{P}_k)$ is defined as:

$$v(\mathcal{P}_k) = u(\mathcal{P}_k) - c(\mathcal{P}_k).$$

Switch operation

Definition

A switch operation $\sigma_{k,\ell}(\mathcal{U})$ is defined as the transfer of players $\mathcal{U}$ from $\mathcal{P}_k$ to $\mathcal{P}_\ell \cup \{\emptyset\}$, $\sigma_{k,\ell}(\mathcal{U}) : \mathcal{P}_k \mapsto \mathcal{P}_k \setminus \mathcal{U}$, and $\mathcal{P}_\ell \mapsto \mathcal{P}_\ell \cup \mathcal{U}$.

Remark 1: if $\mathcal{P}_\ell = \emptyset$, then $\sigma_{k,\ell}(\mathcal{U})$ leads to a new coalition.
Remark 2: if $\mathcal{U} = \mathcal{P}_k$, then $\sigma_{k,\ell}(\mathcal{U})$ leads to the merge of $\mathcal{P}_k$ with $\mathcal{P}_\ell$.

Definition (Switch Operation Gain)

This gain $g(\sigma_{k,\ell}(\mathcal{U}))$ associated with $\sigma_{k,\ell}(\mathcal{U})$ is defined as:

$$g(\sigma_{k,\ell}(\mathcal{U})) := r_{\mathcal{U}}(\mathcal{P}_\ell \cup \mathcal{U}) - r_{\mathcal{U}}(\mathcal{P}_k),$$

with $r_{\mathcal{U}}(\mathcal{S})$ defined as $r_{\mathcal{U}}(\mathcal{S}) := v(\mathcal{S}) - v(\mathcal{S} \setminus \mathcal{U})$ and interpreted as the added value of having players $\mathcal{U}$ in coalition $\mathcal{S}$.

Nash equilibrium

Definition (Preference relation)

This relation $\succeq$ is defined as a complete and transitive binary relation between two switch operations $\sigma_{k,\ell}(\mathcal{U}_i)$ and $\sigma_{k',\ell'}(\mathcal{U}_j)$ such that:

$$\sigma_{k,\ell}(\mathcal{U}_i) \succeq \sigma_{k',\ell'}(\mathcal{U}_j) \Leftrightarrow g(\sigma_{k,\ell}(\mathcal{U}_i)) > g(\sigma_{k',\ell'}(\mathcal{U}_j))$$

In Game Theory, relevant points satisfy equilibrium property:

- No global cost function (if it exists, Game Theory is useless)
- Each player (here, coalition) has its own goal and should find a trade-off/equilibrium with the others
- In non-cooperative game (with rational players), *Nash equilibrium*

Definition (Nash stability for coalition game)

$\Pi = \{\mathcal{P}_1, \dots, \mathcal{P}_K\}$ is Nash-stable if $\forall \mathcal{P}_k$, $\forall P_\ell \in \Pi \cup \{\emptyset\}$, $\forall i \in \mathcal{P}_k$,

$$g(\sigma_{k,\ell}(\{i\}) \leq 0$$

It exists no single node switch operation with a strictly positive gain

Algorithm for unstructured network

- Unstructured traffic based wireless network
- Goal: build stable clusters (even with moving nodes)
- Solution: **gathering nodes with high link capacities between each other**
- As a byproduct, communications within the cluster easier to manage, and then elect a cluster head (as a hub)

Consequently, the coalition reward is the sum capacity of all intra-cluster links:

$$u(\mathcal{P}_k) := \sum_{i \in \mathcal{P}_k} \sum_{j \in \mathcal{P}_k | (i,j) \in \mathcal{E}} \kappa(i,j)$$

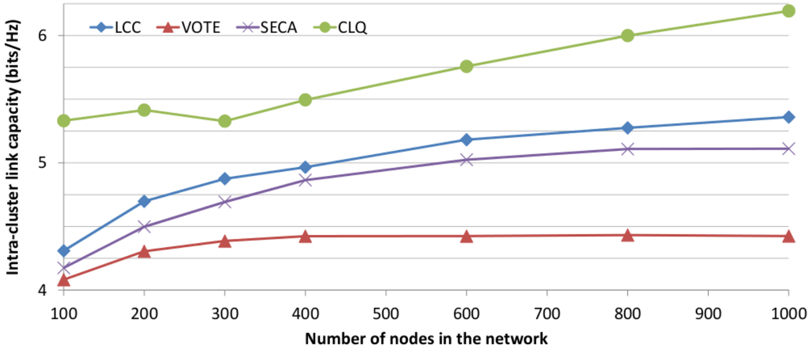
where $\kappa(i,j)$ denotes the *capacity* of link (i,j)

$$\kappa(i,j) = \log_2 \left(1 + \frac{\text{ChannelGain}_{i,j}\text{Power}}{\text{Interference+Noise}} \right).$$

The cost $c(\mathcal{P}_k)$ is 0 if cluster size constraint satisfied and ∞ otherwise

Numerical results

Proposed algorithm: Clustering with Link Quality (CLQ).



Part 3: Attributed graphs

3.1: Node classification

Goal

Predict the class/label of each unlabeled node of the graph by relying

- on nodes' features and
- on nodes' connections within the graph.

Examples:

- in social networks, people are more likely to connect with those who share the same areas of interest
- in research articles' database, more likely to have connections/citations between articles dealing with the same research topic

That's the homophily principle

Information taken into account

- Label Propagation (LP): propagated through the adjacent nodes at each step. Requires algorithms for merging labels' information
- Feature propagation (FP): propagated through the adjacent nodes at each step. Requires algorithms for merging features

FP approach

Main idea: weighted averaging of the current features of adjacent nodes (sometimes followed by a nonlinear function)

- Graph neural networks (GNN): Neural Networks adapted to the attributed graphs. Training done with labeled nodes
- **Our contribution:** we derive in *closed-form* a classifier
    - interpretable algorithm (no black box)
    - less complex since no training
    - No embedding (as done by GNN)

Example

- p: average probability of intra-class connection
- q: average probability of inter-class connection

- Degree of impurity (DoI) of the graph is $\frac{q}{p}$
- DoI $\ll 1$ leads to graph with communities (nodes with similar features are connected to each other).

	Cora	**Citeseer**
Intra-class connectivity (p)	23×10^{-3}	12×10^{-3}
Inter-class connectivity (q)	5.5×10^{-3}	4.3×10^{-3}
Degree of Impurity (q/p)	0.23	0.36
Logistic Regression (LR)	56.0%	57.2%
Two-layer GNN	81.5%	70.3%
Gain between GNN and LR	+45.5%	+22.9%

Problem statement

Classifier based on Bayesian decision theory: Maximum A Posteriori

- $\mathcal{V}_u$: set of nodes involved in the classification of node u.
- $\mathcal{X}_u = \{\boldsymbol{x}_u\} \cup \{\boldsymbol{x}_v, v \in \mathcal{V}_u\}$: set of features of node u and its "helping" nodes
- y_u: class of node u (what we are looking for!)
- D_k: probability density function of features belonging to class k. For any u,

$$D_k(\boldsymbol{x}_u) = p(\boldsymbol{x}_u | y_u = k).$$

Graph-Assisted Bayesian (GAB) Classifier

$$\hat{k}_u = \arg\max_k P_u(k)$$

with $P_u(k) = P(y_u = k | \mathcal{X}_u, \mathcal{I}_\mathcal{G})$, and by knowing information on $\mathcal{X}_u$ and on the graph $\mathcal{I}_\mathcal{G}$ (e.g., its partial connectivity through the set $\mathcal{V}_u$)

Problem solution

- $\arg\max$: here no complexity issue since small amount of classes
- Derivations of $P_u(k)$. Bayes' rule

$$P_u(k) = \frac{P(\mathcal{X}_u|y_u = k, \mathcal{I}_\mathcal{G})P(y_u = k|\mathcal{I}_\mathcal{G})}{P(\mathcal{X}_u|\mathcal{I}_\mathcal{G})} \propto Q_u(k)\pi_k$$

with $\pi_k = P(y_u = k|\mathcal{I}_\mathcal{G})$ a priori classes' probability

Let Δ_u be the diameter of the set $\mathcal{V}_u$.

$$Q_u(k) = D_k(\boldsymbol{x}_u)\prod_{d=1}^{\Delta_u}\prod_{v\in\mathcal{N}_u(d)}\left(\sum_{k'=1}^{K} r_{u,v}(k,k')D_{k'}(\boldsymbol{x}_v)\right)$$

with $r_{u,v}(k,k') = p(y_v = k'|y_u = k, \mathcal{I}_\mathcal{G})$ the probability to be on class k' for node v given the fact that we are in class k for node u and we have information on the graph $\mathcal{I}_\mathcal{G}$.

Example: $\mathcal{V}_u = \{v\}$, known $k_v = 1$, $\pi_1 = \pi_2 = 1/2$, and $\Delta_u = 1$:

$$Q_u(1) = D_1(\boldsymbol{x}_u)\frac{p}{p+q} \text{ and } Q_u(2) = D_2(\boldsymbol{x}_u)\frac{q}{p+q}$$

Main result

Assumptions

- 2 equilikely classes
  - $p(k)$ probability that two nodes from class k are connected
    p average of $\{p(k)\}_k$
  - q probability that two nodes from different classes are connected.
- Information on graph is 1-hop

We get

$$\begin{array}{c|c} r(1,2) = \frac{q}{p(1)+q} & r(2,2) = \frac{p(2)}{q+p(2)} \\ \hline r(1,1) = \frac{p(1)}{p(1)+q} & r(2,1) = \frac{q}{q+p(2)} \end{array}$$

GAB does not depend on the graph iff $r(1,2) = r(2,2)$ and $r(1,1) = r(2,1)$

$$q = \sqrt{p(1)p(2)} = \overline{p}_{\text{geometric}}$$

iff

$$\text{DoI} = \frac{\overline{p}_{\text{geometric}}}{\overline{p}_{\text{arithmetic}}} \leq 1$$

Graph Neural Network (1/2)

Main idea

- Use graph structure in addition to node and edge features to generate node representation vectors (i.e., embedding)
- Aggregate the features of neighboring nodes and edges
- Output of the ℓ-th layer of GNN is

$$\boldsymbol{h}_u^{(\ell)} = \sigma^{(\ell)}(\phi^{(\ell)}(\boldsymbol{h}_u^{(\ell-1)}, \{\boldsymbol{h}_v^{(\ell-1)} : v \in \mathcal{N}_u\}))$$

  where
  - $\boldsymbol{h}_u^{(\ell)}$ representation vector of node u at ℓ-th layer ($\boldsymbol{h}_u^{(0)} = \boldsymbol{x}_u$)
  - $\sigma^{(\ell)}$ activation function
  - $\phi^{(\ell)}$ linear function associated with weights' matrix $\mathbf{W}^{(\ell)}$
  - First-order GNN has 1 layer (1-hop neighborhood in the graph)

- Usually, activation function is a rectified linear unit (ReLU).
- For the last layer, $\mathrm{softmax}$ which provides a probability
- Then node u is attributed to the class with the highest probability

Graph Neural Network (2/2)

Graph Convolutional Neural Network (GCN):

$$\phi_u^{(\ell)} = \mathbf{W}^{(\ell)} \left(\frac{\boldsymbol{h}_u^{(\ell-1)}}{d_u + 1} + \sum_{v \in \mathcal{N}_u} \frac{\boldsymbol{h}_v^{(\ell-1)}}{\sqrt{(d_u+1)(d_v+1)}} \right)$$

Graph Isomorphism Network (GIN):

$$\phi_u^{(\ell)} = \mathbf{W}^{(\ell)} \left((1+\alpha)\boldsymbol{h}_u^{(\ell-1)} + \sum_{v \in \mathcal{N}_u} \boldsymbol{h}_v^{(\ell-1)} \right)$$

Graph convolution Operator Network (GON):

$$\phi_u^{(\ell)} = \mathbf{W}_1^{(\ell)} \boldsymbol{h}_u^{(\ell-1)} + \mathbf{W}_2^{(\ell)} \Big(\sum_{v \in \mathcal{N}_u} \boldsymbol{h}_v^{(\ell-1)} \Big)$$

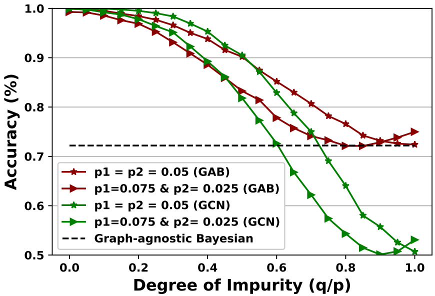
Graph Attention Network (GAT):

$$\phi_u^{(\ell)} = \sum_{v \in \mathcal{N}_u \cup \{u\}} \alpha_{u,v}^{(\ell)} \mathbf{W}^{(\ell)} \boldsymbol{h}_v^{(\ell-1)} \text{ with } \alpha_{u,v}^{(\ell)} \propto e^{\varsigma(\mathbf{w}^{(\ell)}[\mathbf{W}^{(\ell)}\boldsymbol{h}_u^{(\ell-1)} \| \mathbf{W}^{(\ell)}\boldsymbol{h}_v^{(\ell-1)}])}$$

the so-called normalized attention coefficients

Numerical illustrations (1/2)

- 2 classes
- Gaussian distributions with different means and covariance matrices
- Number of nodes $N = 5,000$ and number of features $F = 500$
- 500 (already-labeled) nodes



- GAB more robust to DoI than GCN
- GCN becomes worse than graph-agnostic (too confident)

Numerical illustrations (2/2)

	Parameters to estimate in GAB	Weights to learn in GNN
Cora	10,087	369,066
PubMed	1,512	129,286

	MLP	**GCN**	**SAGE**	**GAT**	**GMN**	**DGCN**	**GBPN**	**GAB**
Cora	72.1	87.1	86.9	87.1	86.4	**87.2**	86.4	86.9
CiteSeer	71.2	73.5	73.5	73.1	72.9	73.9	74.8	**75.2**
PubMed	86.5	87.1	87.8	88.1	86.7	84.7	**88.5**	86.4
CS	94.2	93.2	93.7	94.0	93.3	94.9	**95.5**	94.5
Physics	95.8	96.1	96.3	96.3	96.1	96.7	**96.9**	96.4

- GAB close to GBPN and GAT, the best ones in the literature
- But interpretability
- But low-complexity

3.2: Graph comparison

Question

- Are two graphs close to each other ?
- Useful in many applications: link prediction, time-varying analysis, etc

Main issues:

- Balance between features and edges ?
- Even if no features, what does it mean two close graphs ?
  - counter-example: by cutting a few edges, new graph is not connected : is it far or not from the original one
  - so just comparing $\mathbf{A}$ is not enough: induced properties are crucial

Detour by the optimal transport

Original problem [Monge1781]

How moving a sand pile with shape 1 into a shape 2 by minimizing the energy consumption ?

Shape : f where $f(x)$ provides the level of sand at x

- $f(x) \geq 0$, and $\int f(x)dx = 1$: probability density function (pdf)

Transport problem

- Transport map: $y = T(x)$
- Transport cost: $c(x, T(x))$, and $C(T) = \int c(x, T(x)) f_1(x)dx$
- Transport application: $T_\#$

$$f_1(\{x : T(x) \in \Omega\}) = f_2(\Omega) \Leftrightarrow T_\# f_1(\Omega) = f_1(T^{(-1)}(\Omega)) = f_2(\Omega)$$

$$T^\star = \arg \min_{T, T_\# f_1 = f_2} C(T)$$

In general, to hard to solve

Relaxation [Kantorovitch1942]

Modification of definition of Transport Map T

- it is not a function anymore
- it is a probability function: given the sand at x, it can be spread at several new positions.

$$x \mapsto T_x$$

$$C(T) = \int \left(\int c(x,y) T_x(y) dy \right) f_1(x) dx$$

s.t.

- Accurate final shape: $f_2(\Omega) = \int (\int_\Omega T_x(y) dy) f_1(x) dx$
- Take only the original shape: $f_1(\Omega) = \int (\int_\Omega T_x(y) f_1(x) dx) dy$

Then consider $T_x(y) f_1(x) = \pi(x,y)$

Relaxation [Kantorovitch1942]

Modification of definition of Transport Map T

$$\pi^\star = \arg\min_\pi \iint c(x,y)\pi(x,y)dxdy$$

s.t.

- $f_2(\Omega) = \int_{y\in\Omega} \left(\int \pi(x,y)dx\right) dy$
- $f_1(\Omega) = \int_{x\in\Omega} \left(\int \pi(x,y)dy\right) dx$

Much easier : Linear programming

Wasserstein distance

Consider two probability mass function (pmf) : discrete version of pdf

- f_1: $\sum_{i=1}^{m} a_i \delta_{x_i}$ (**a** non-negative vector summing to 1)
- f_2: $\sum_{i=1}^{n} b_i \delta_{y_i}$ (**b** non-negative vector summing to 1)

p-Wasserstein distance

$$W_p(f_1, f_2) = \min_{\{\gamma_{i,j}\}_{i,j}} \sum_{i=1}^{m} \sum_{j=1}^{n} |x_i - y_j|^p \gamma_{i,j}$$

s.t.

- $b_j = \sum_{i=1}^{m} \gamma_{i,j}, \ \forall j$
- $a_i = \sum_{j=1}^{n} \gamma_{i,j}, \ \forall i$

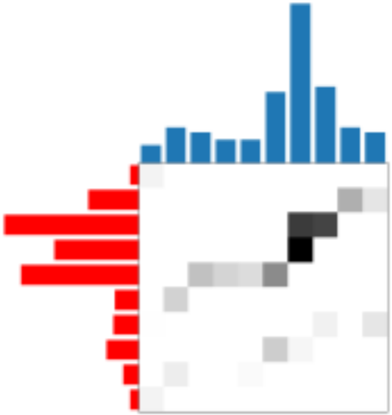
where $\gamma_{i,j}$ is the quantity of material going from x_i to y_j

Remark: Completely different from standard Kullback-Leibler distance (related to entropy)

$$D(f_1||f_2) = \int \log_2 \left(\frac{f_1(x)}{f_2(x)} \right) f_1(x) dx$$

Example

- Blue: Source distribution
- Red: Target distribution



Graph Diffusion Distance

- Non-attributed graph
- related to Heat diffusion
- Idea: similar graph will diffuse in the same way the heat

$$\mathrm{GDD} = \max_{\tau \geq 0} \left\| \exp\left(-\tau \mathbf{L}_1\right) - \exp\left(-\tau \mathbf{L}_2\right) \right\|_2^2$$

where $\|.\|_f^2$ is the Frobenius square norm (summing the square of each matrix component)

Gromov-Wasserstein distance

- Non-attributed graph
- Adapted to Graph
- Matrices $C^s \in \mathbb{R}^{m \times m}$ and $C^t \in \mathbb{R}^{n \times n}$

$$\mathrm{GW} = \min_{\{\gamma_{i,j}\}_{i,j}} \sum_{i,i',j,j'} d(C^s_{i,i'}, C^t_{j,j'})\gamma_{i,j}\gamma_{i',j'}$$

s.t.

- $a_i = \sum_{j=1}^{n} \gamma_{i,j}, \ \forall i$
- $b_j = \sum_{i=1}^{m} \gamma_{i,j}, \ \forall j$

Application to Graph:

- **C** may be the adjacency matrix **A**
- **C** may be a similarity matrix between nodes
- Hyperparameters **a** and **b** to be tuned

Fused Gromov-Wasserstein distance

- Attributed graph
- Matrices $C^s \in \mathbb{R}^{m\times m}$ and $C^t \in \mathbb{R}^{n\times n}$

$$\mathrm{FGW} = \min_{\{\gamma_{i,j}\}_{i,j}} \sum_{i,i',j,j'} \left[(1-\alpha)d_1(\boldsymbol{x}_i, \boldsymbol{x}_j)\gamma_{i,j} + \alpha d_2(C^s_{i,i'}, C^t_{j,j'})\gamma_{i,j}\gamma_{i',j'}\right]$$

s.t.

- $a_i = \sum_{j=1}^n \gamma_{i,j}, \ \forall i$
- $b_j = \sum_{i=1}^m \gamma_{i,j}, \ \forall j$

Diffusion-Wasserstein distance

- Attributed graph
- $\mathbf{Y}^s = \exp(-\tau^s \mathbf{L}_s).\mathbf{X}^s$ heat diffusion with initial values $\mathbf{X}^s$
- $\mathbf{Y}^t = \exp(-\tau^t \mathbf{L}_t).\mathbf{X}^t$ heat diffusion with initial values $\mathbf{X}^t$

$$\mathrm{DW}_{\tau^s,\tau^t} = \min_{\{\gamma_{i,j}\}_{i,j}} \sum_{i,j} d_1(\boldsymbol{y}_i^s, \boldsymbol{y}_j^t)\gamma_{i,j}$$

s.t.

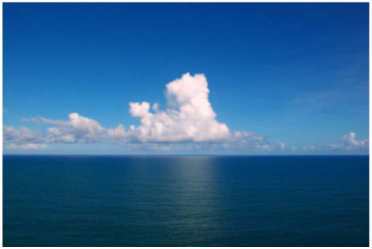
- $a_i = \sum_{j=1}^{n} \gamma_{i,j}, \ \forall i$
- $b_j = \sum_{i=1}^{m} \gamma_{i,j}, \ \forall j$

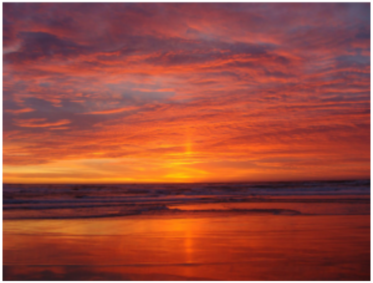
Extreme cases:

- $\tau^s = \tau^t = 0$, Wasserstein distance
- $\tau^s = \tau^t = \infty$, average comparison of features

Numerical illustrations (1/3)

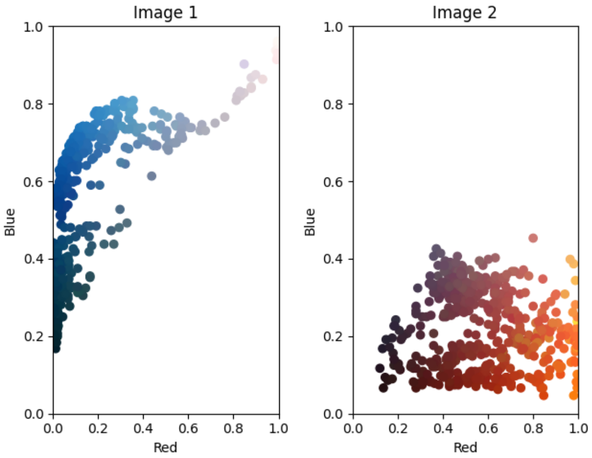
Image color adaptation


Original image


Target color

Numerical illustrations (2/3)

Optimal transport on color distribution



Numerical illustrations (3/3)

Part 4: From graphs to vectors

Embedding

Main idea

- Vector : nice representation for signals
- Why? many algorithms adapted to vectors
  - in classification (k-means, NN with vector as input)
  - in regression (linear, NN with vector as input)

Embedding:

- representing any type of signal as a vector
- practical and efficient but not necessary smart (see 3.1)

Examples:

- Text: word2vec
  - close vector $=$ synonym
  - semantic vector space: $v_{queen} + v_{man} = v_{king}$
- Graph: graph representation learning (through GNN)
  - “close” points in graph are close points in vector space
  - But, what does it mean “close” in graph when trade-off between edges and features

Representation self-learning

- $\boldsymbol{x}_u \in \mathbb{R}^P$ feature vector at node u
- $\mathbf{X} \in \mathbb{R}^{N \times P}$: matrix stacking initial feature vectors of all nodes.
- $\mathbf{A}$: adjacency matrix of the graph

Goal

- Self-learning node representation (without human annotation/tag)
- i.e., learning a graph neural network (with L layer) encoder f

$$\mathbf{H}^{(L)} := f(\mathbf{X}, \mathbf{A}) \in \mathbb{R}^{N \times P'}$$

  with
  - $P' \leq P$ the embedding size
  - $u-$th row of $\mathbf{H}^{(L)}$ is the embedding/representation vector $\boldsymbol{h}_u^{(L)}$ of node u.
- Finding a appropriate *criterion* to optimize f

Contrastive learning

Given a feature $\boldsymbol{h}_u$ of node u, we generate

- a *positive* example $\boldsymbol{h}_u^+$ (close to $\boldsymbol{h}_u$)
- a set of *negative* examples Q_u

We define a loss $\mathcal{L}$ offering low value when $\boldsymbol{h}_u$

- similar to $\boldsymbol{h}_u^+$
- dissimilar to all elements $\boldsymbol{h}^-$ of Q_u

A standard loss

$$\mathcal{L} = -\sum_{u\in\mathcal{G}} \boldsymbol{h}_u^{\mathrm{T}}\boldsymbol{h}_u^+ + \log\sum_{u\in\mathcal{G}}\left(\exp(\boldsymbol{h}_u^{\mathrm{T}}\boldsymbol{h}_u^+) + \sum_{\boldsymbol{h}^-\in Q_u}\exp(\boldsymbol{h}_u^{\mathrm{T}}\boldsymbol{h}^-)\right)$$

How generating negative examples?

- **Feature-based sampling:**
  - based on comparisons between nodes' intrinsic features.
  - For node *u*, we consider as negatives all nodes *v* whose intrinsic features are neither too close nor too far from those of node *u*

$$\frac{\boldsymbol{x}_u^{\mathrm{T}}\boldsymbol{x}_v}{\|\boldsymbol{x}_u\|\|\boldsymbol{x}_v\|} \in [\omega_{LB}, \omega_{UB}]$$

    Negatives are different from positive but quite hard to distinguish from the current sample

- **Graph-based sampling:**
  - based on graph structure
  - For node *u*, we consider as negatives all nodes *v* located at ℓ-hop of node *u*

Algorithm

- Consider two (small) stochastic perturbations t_1 and t_2 on edges and features
  - $(\mathbf{X}_1, \mathbf{A}_1) \sim t_1(\mathbf{X}, \mathbf{A})$
  - $(\mathbf{X}_2, \mathbf{A}_2) \sim t_2(\mathbf{X}, \mathbf{A})$
- Apply the current encoder to exhibit the node representations
  - the baseline representation $\mathbf{H}^L = f(\mathbf{X}_1, \mathbf{A}_1)$
  - the positive example $\mathbf{H}^L_+ = f(\mathbf{X}_2, \mathbf{A}_2)$
- Select negative examples
  - Features-based sampling
  - Connection-based sampling
- Update weights of f using the loss function $\mathcal{L}$

Numerical illustrations

- Test the node representation into a classification problem
- Once embedding done, classification relying on logistic regression applied
- Embedding size $P' = 512$

	Cora	**Citeseer**	**Pubmed**	**Arxiv**
Raw features	47.9	49.3	69.1	55.5
DeepWalk	67.2	43.2	65.3	70.1
DeepWalk + features	70.7	51.4	74.3	-
EP-B	78.1	71.0	79.6	68.0
DGI	82.3	71.8	76.8	**70.2**
Proposed Method	**83.6**	**72.5**	**79.8**	**70.2**
GCN (supervised)	81.5	70.3	79.0	71.7

Conclusion

Graph is a fascinating mathematical structure

- using different mathematical branches
- related to practical problems
- but difficult to manage

Problems not treated here

- Epidemic propagation (rumor spreading, max-consensus)
- Link prediction
- Random Graph
- Structured Graph (like in Chemistry)
- Graph drawing

Our publications devoted to graphs

PhD theses:

- Hakim Hafidi (co-supervised with Mounir Ghogho (UIR, Morocco)), “Robust Machine Learning for Graphs”, Feb. 2023
- Raphaël Massin (co-supervised with Christophe Le Martret (Thales)), “On distributed node clustering in mobile ad hoc networks”, Nov. 2016
- Franck Iutzeler (co-supervised with Walid Hachem), “Distributed estimation and optimization in asynchronous networks”, Dec. 2013

Other references

- D. Spielman, “Spectral and Algebraic Graph Theory”, Booklet, Yale University, 2019
- A.-L. Barabasi, “Network Science”, Cambridge Press University, 2016
- A. Ortega, P. Frossard, J. Kovacevic, J. Moura, P. Vandergheynst, “Graph signal processing: Overview, challenges, and applications”, Proceedings of the IEEE, 2018
- M. Newman, “Fast algorithm for detecting community structure in networks,” 2003
- S. Ferradans, N. Papadakis, G. Peyre, J. Aujol, “Regularized discrete optimal transport”, SIAM Journal on Imaging Sciences, 2014
- L. Brogat-Motte, R. Flamary, C. Brouard, J. Rousu, F. d’Alché-Buc, “Learning to Predict Graphs with Fused Gromov-Wasserstein Barycenters”, ICML, 2022