Graphs for data science and ML

Machine Learning for graphs and with graphs

P. Borgnat, CNRS, LP ENSL **(5)**

Graph Laplacian: An Analogy for a Graph FT

A fundamental analogy

On *any* graph, the **eigenvectors χ_i of the Laplacian matrix L will be considered as the Fourier modes**, and its eigenvalues λ_i the associated (squared) frequencies.

Hence, a Graph Fourier Transform is defined as:

$$\boxed{\hat{x} = \chi^\top x}$$

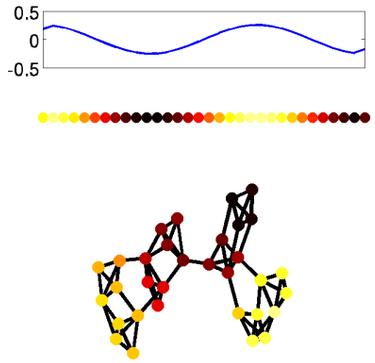
where $\chi = (\chi_0 | \chi_1 | \dots | \chi_{N-1})$.

- Two ingredients:
  - **Fourier modes** = Eigenvectors χ_i (with increasing oscillations)
  - **Frequencies** = The measures of variations of an eigenvector is linked to its eigenvalue:

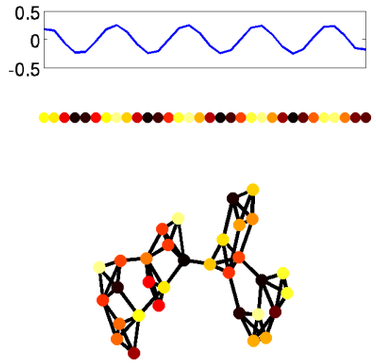
$$\frac{||\nabla \chi_i||^2}{||\chi_i||^2} = \lambda_i$$

because: $\forall \mathbf{x} \in \mathbb{R}^N \quad \sum_{e=(i,j)\in E} A_{ij}(\mathbf{x}_i - \mathbf{x}_j)^2 = \mathbf{x}^\top \mathbf{L}\mathbf{x}$ is the Dirichlet norm

LOW FREQUENCY:



HIGH FREQUENCY:



Graph Laplacian: An Analogy for a Graph FT

[Tremblay, Gonçalves, PB, 2017]

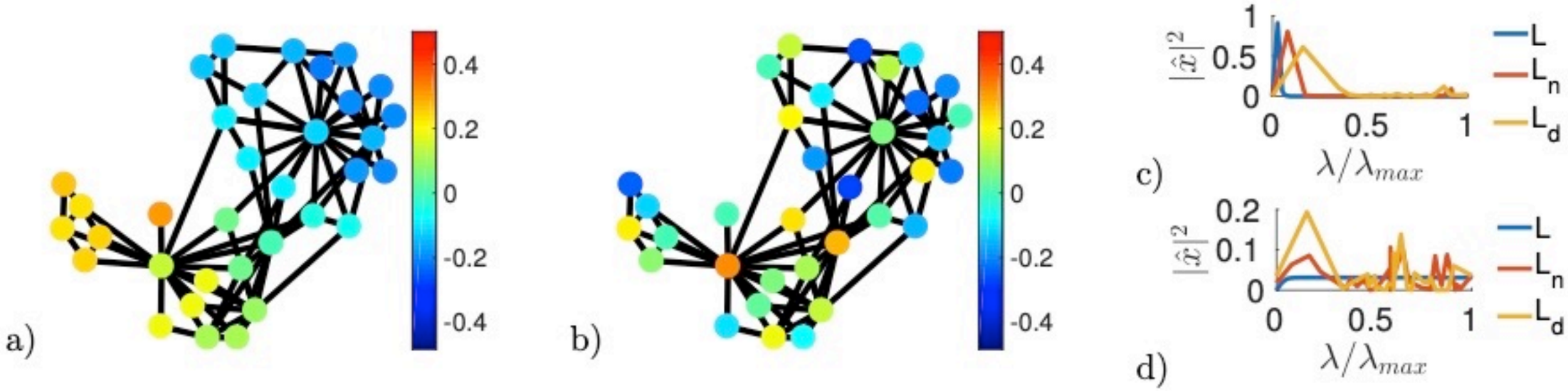


Figure 1: ***Two graph signals and their GFTs.*** *Plots a) and b) represent respectively, a low-frequency and a high-frequency graph signal on the binary Karate club graph [21]. Plots c) and d) are their corresponding GFTs computed for three reference operators:* $\mathbf{L}$, $\mathbf{L_n}$ *and* $\mathbf{L_d}$ *(equivalent to the GFT defined via the adjacency matrix).*

Use GSP to process data
which are on graphs,
or which are the graphs!

(1) Filters on graphs

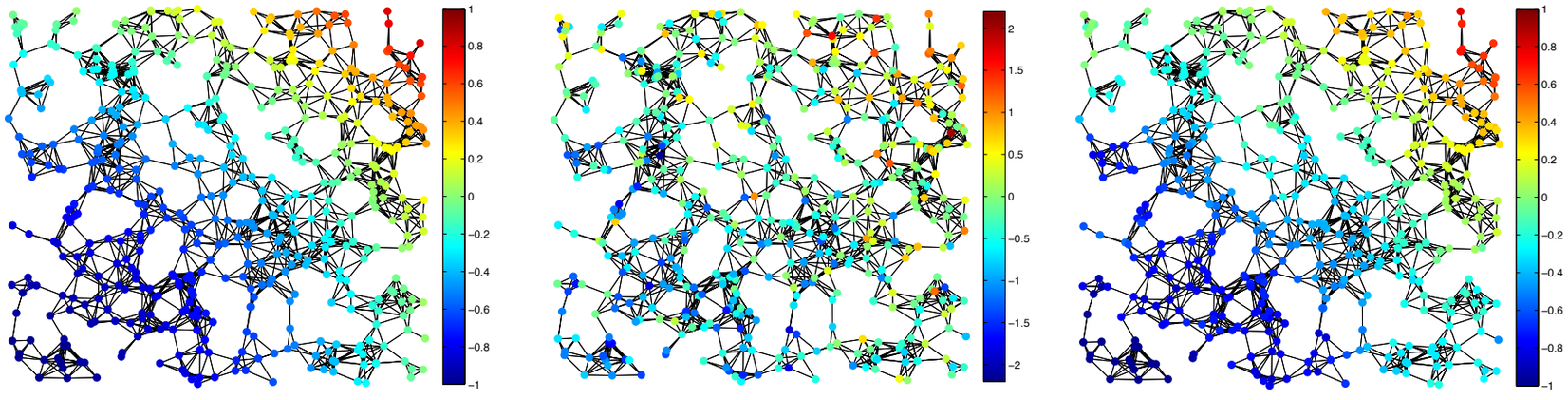
Example 1: Recovery of signals on graphs

- Denoising of a graph signal, when observing $y = x_0 + \epsilon$, formulated as an inverse problem:

$$x_* = \arg\min_x ||x - y||_2^2 + \gamma x^\top L x$$

because remember that : $x^\top L x = \sum_{e=(i,j)\in E} A_{ij}(x_i - x_j)^2$

- Graph-Fourier coefficients: $\hat{x} = \chi^\top x$
- Solution: $\hat{x_*}(i) = \frac{1}{1+\gamma\lambda_i}\hat{y}(i)$ (1st-order "low pass" filter)



Original | Noisy | Denoised

[P. Vandergheynst, EPFL, 2013]

(1) Filters on graphs

Filtering

Definition of graph filtering

We define a linear filter $\mathcal{H}$ by its function h in the Fourier domain.
It is discrete and defined on the eigenvalues $\lambda_i \to h(\lambda_i)$.

$$\widehat{\mathcal{H}(x)} = \begin{pmatrix} h(\lambda_0)\,\hat{x}(0) \\ h(\lambda_1)\,\hat{x}(1) \\ h(\lambda_2)\,\hat{x}(2) \\ \dots \\ h(\lambda_{N-1})\,\hat{x}(N-1) \end{pmatrix} = \hat{\boldsymbol{H}}\,\hat{x} \text{ with } \hat{\boldsymbol{H}} = \begin{pmatrix} h(\lambda_0) & 0 & 0 & \dots & 0 \\ 0 & h(\lambda_1) & 0 & \dots & 0 \\ 0 & 0 & h(\lambda_2) & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & h(\lambda_{N-1}) \end{pmatrix}$$

In the node-space, the filtered signal $\mathcal{H}(x)$ can be written:

$$\mathcal{H}(x) = \chi\,\hat{\boldsymbol{H}}\,\chi^\top\,x$$

In term of calculus of operator on a graph, this reads

$$\mathcal{H}(x) = h(L)\cdot x$$

- Alternative definition: operator **H** that commutes with the reference operator, here the Laplacian (yet could be some “shift”, e.g. **A**)
- Parametric formulation: $h(L) = \sum_{k=0}^{K} h_k L^k$
  (leads to ARMA filters; to distributed implementations)

(1) Filters on graphs

Example [Tremblay, Gonçalves, PB, 2017]

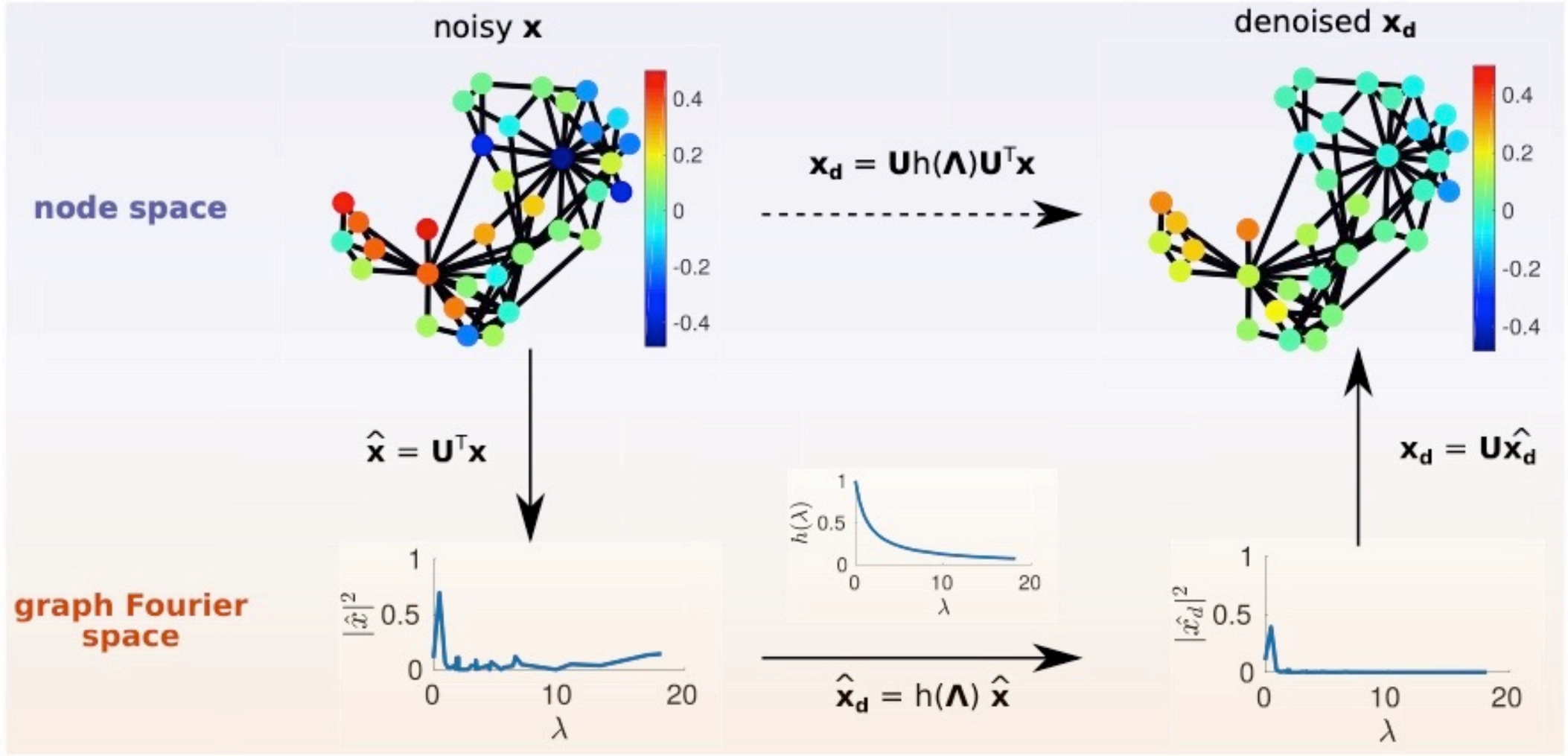


Figure 3: ***Illustration of graph filters: a denoising toy experiment.*** *The input signal* $\mathbf{x}$ *is a noisy version (additive Gaussian noise) of the low-frequency graph signal displayed in Fig. 1. We show here the filtering operation in the graph Fourier domain associated to* $\mathbf{R} = \mathbf{L_n}$.

Diffusion on graphs

Functional calculus on graph

Objective: define the effect of function on graph data

We use the simple property that $\underline{\underline{L}}^n \underline{\chi}_k = \lambda_k^n \underline{\chi}_k$

Then, for any polynomial function f, we have $f(\underline{\underline{L}}) = \sum_{\lambda_k \in Sp(\underline{\underline{L}})} f(\lambda_k) \underline{\chi}_k \underline{\chi}_k^T$

$= \underline{\underline{\chi}}\, f(\underline{\underline{\Lambda}})\, \underline{\underline{\chi}}^T$

Using approximation theorem, it holds for any function.

Example: define a **diffusive** process on a graph

With the analogy: $f(u,t)$ is a diffusion if it follows $\boxed{\frac{\partial f}{\partial t} = -\underline{\underline{L}}\, f}$

Applying the GFT: $\frac{\partial}{\partial t} \tilde{f}(\lambda_k, t) = -\lambda_k \tilde{f}(\lambda_k, t)$

hence, if $f(u, t=0) = f_0(u)$, we have $\tilde{f}(\lambda_k, t) = e^{-t\lambda_k} \tilde{f}_0(\lambda_k)$

With fctal calculus: $\boxed{\underline{f}(t) = e^{-t\underline{\underline{L}}}\, \underline{f}_0}$

Explicit expression: $f(u,t) = \sum_k e^{-t\lambda_k} \tilde{f}_0(\lambda_k)\, \chi_k(u)$

This acts as a filter $e^{-t\lambda_k}$ on the GFT of the initial condition $\underline{f}_0$.

Diffusion on graphs (2) — Illustration

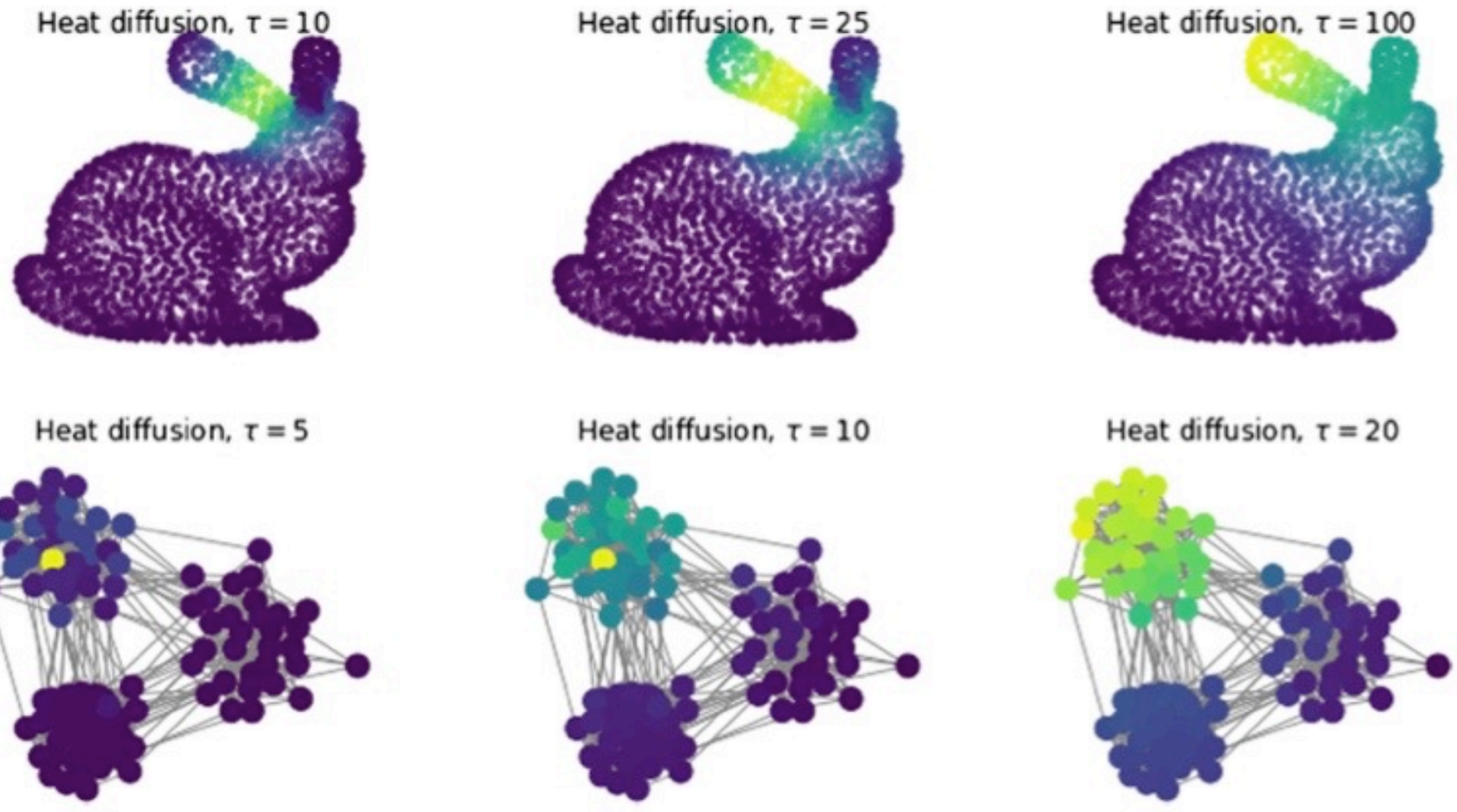


Fig. 1. Illustration of the heat diffusion over a 2-d manifold (top), and over a graph with communities (bottom), at different time τ. In both graphs, the heat spreads from node to node, following the edges. Top: the initial hot spot is a node located on the ear of the bunny. The Bunny graph is a discretization of a 2-d surface, with nodes connected to their nearest neighbours in 3 d. Bottom: The diffusion starts inside a community and quickly spreads within it.

[Ricaud et al., 2019]

Diffusion on graphs (3) — Use to define a distance between graphs

- from [Hammond, Gur, Johnson, GlobalSIP 2013] "**GRAPH DIFFUSION DISTANCE**: A DIFFERENCE MEASURE FOR WEIGHTED GRAPHS BASED ON THE GRAPH LAPLACIAN EXPONENTIAL KERNEL" (Title way too long!)
- They define a **Diffusion distance between graphs** having the same number of nodes

$$\xi(A_1, A_2; t) = \sum_{i,j} ((\exp(-tL_1))_{i,j} - (\exp(-tL_2))_{i,j})^2$$
$$= ||\exp(-tL_1) - \exp(-tL_2)||_F^2 \qquad (2)$$

$$d_{gdd}(A_1, A_2) = \max_t \sqrt{\xi(A_1, A_2; t)}.$$

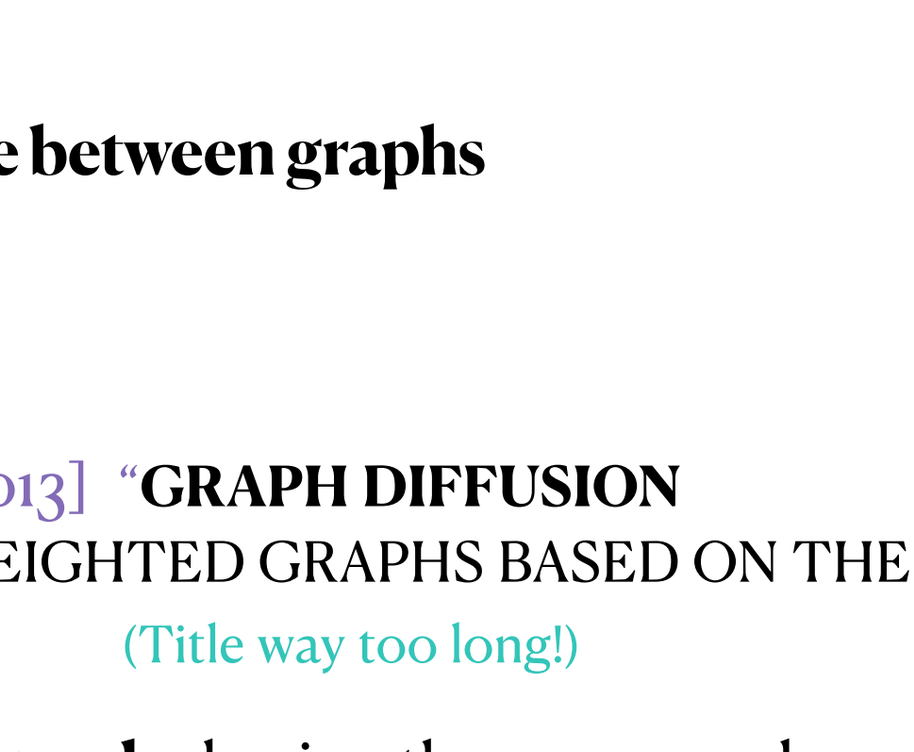


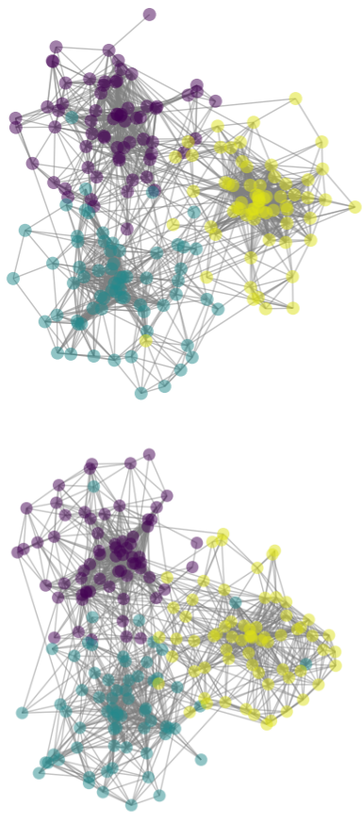
Fig. 1. (a) Barbell graph, and single-edge perturbations, for $N = 5$, $K = 2$. (b) Plot of ratio $d_{gdd}(G^{N,2}, G^{N,2}_{br})/d_{gdd}(G^{N,2}, G^{N,2}_{cc})$ vs N. (c) Plot of $\xi(t)$ for $A_1 = G^{5,2}$, $A_2 = G^{5,2}_{cc}$, red dot indicates maximum, corresponding to $d_{gdd}(A_1, A_2)^2$. (d) Values of normalized edge deletion perturbation, on edges of $G^{5,2}$.

Graph Diffusion Wasserstein Distances & Application to Domain Adaptation for Graphs

From Amélie Barbe PhD thesis (12/2021) ; ECML-PKDD 2020 ;

GRETSI 2019 ; ICTAI 2021 ; GRETSI (2019) ; ICASSP 2022

Joint work with Marc Sebban (LabHC; Saint-Etienne) ; Rémi Gribonval, Paulo Gonçalves, and Titouan Vayer (LIP, Inria, ENS de Lyon) ; Sybille Marcotte (now PhD candidate in Paris)



$$X^s \xrightarrow{\exp(-\tau^s L^s)\cdot} \tilde{X}^s \searrow$$

$$X^t \xrightarrow{\exp(-\tau^t L^t)\cdot} \tilde{X}^t \nearrow \quad \tilde{M} \xrightarrow{\min_{\gamma \in \Pi(a,b)} \left\{ \langle \gamma, \tilde{M}^p \rangle_F \right\}} \mathrm{DW}_p^p(U^s, U^t)$$

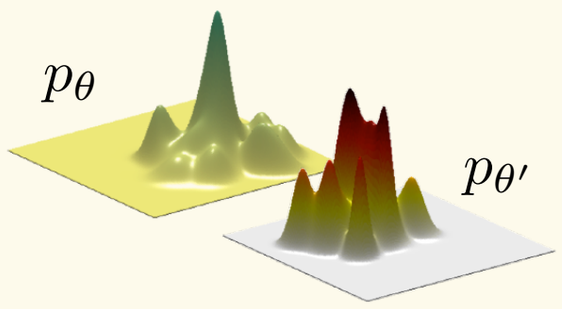
A parenthesis in advance — Optimal Transport: a generic tool to probe the geometry of probability measures

- **Optimal Transport**: an approach to **compute a distance between 2 distributions**, while **finding the optimal coupling (or transport plan)** between them
- Put forward in Data Science/Processing & ML since…
  - since ~2000 in image processing (Earth Mover Distance); well before in mathematics (cf. [Villani, 2003]); in the 70's for the Mallows distance in statistics,…
    - *(see my completely ignored ICASSP paper of 2012: "Using Surrogates and Optimal Transport for Synthesis of Stationary Multivariate Series […]")* (Title way too long!)
- cf. "Computational Optimal Transport" G. Peyré & M. Cuturi, 2019

  https://arxiv.org/abs/1803.00567v4
- cf. "A primer on Optimal Transport", Cuturi & Salomon, NIPS 2017 Tutorial

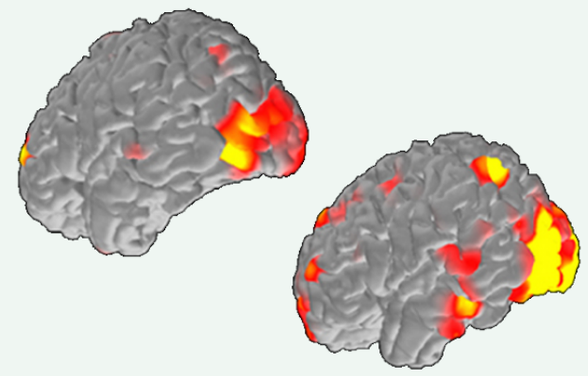
  https://optimaltransport.github.io/slides/ (and other resources)
- **cf. Titouan Vader's Lectures at the end of this course !**

Optimal Transport: a generic tool to probe the geometry of probability measures



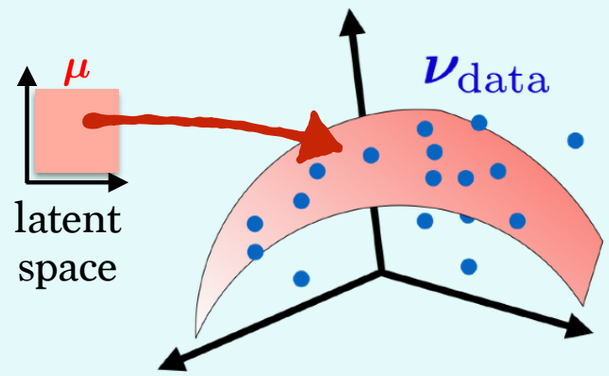
Statistical Models

Bags of features



Brain Activation Maps

Generative Models vs. data



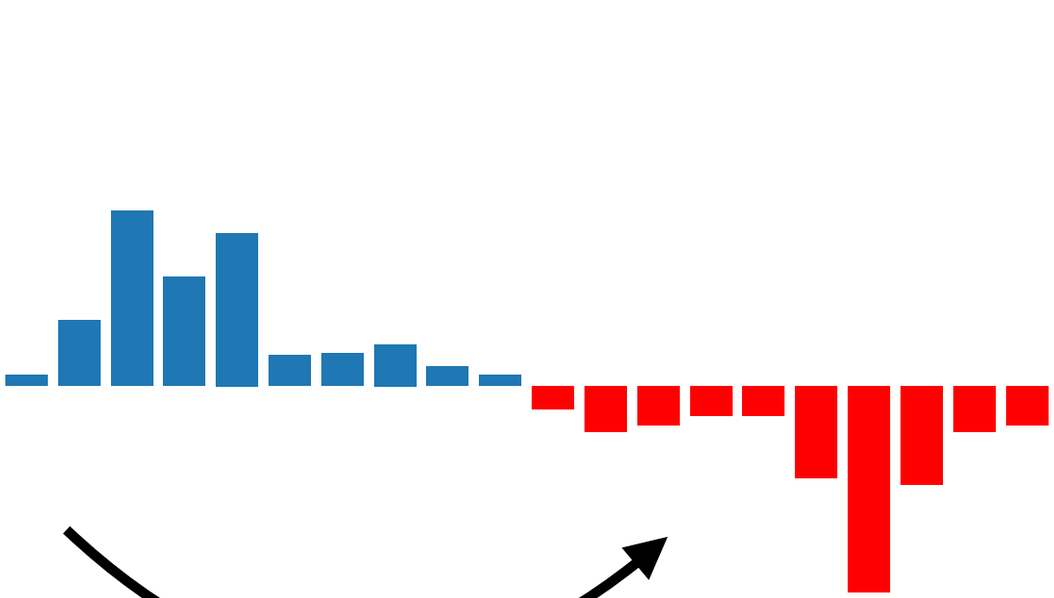
Color Histograms

- from Cuturi & Salomon "A primer on Optimal Transport", NIPS 2017 Tutorial

Optimal Transport for distributions

- from "Computational Optimal Transport" (G. Peyré & M. Cuturi), 2019

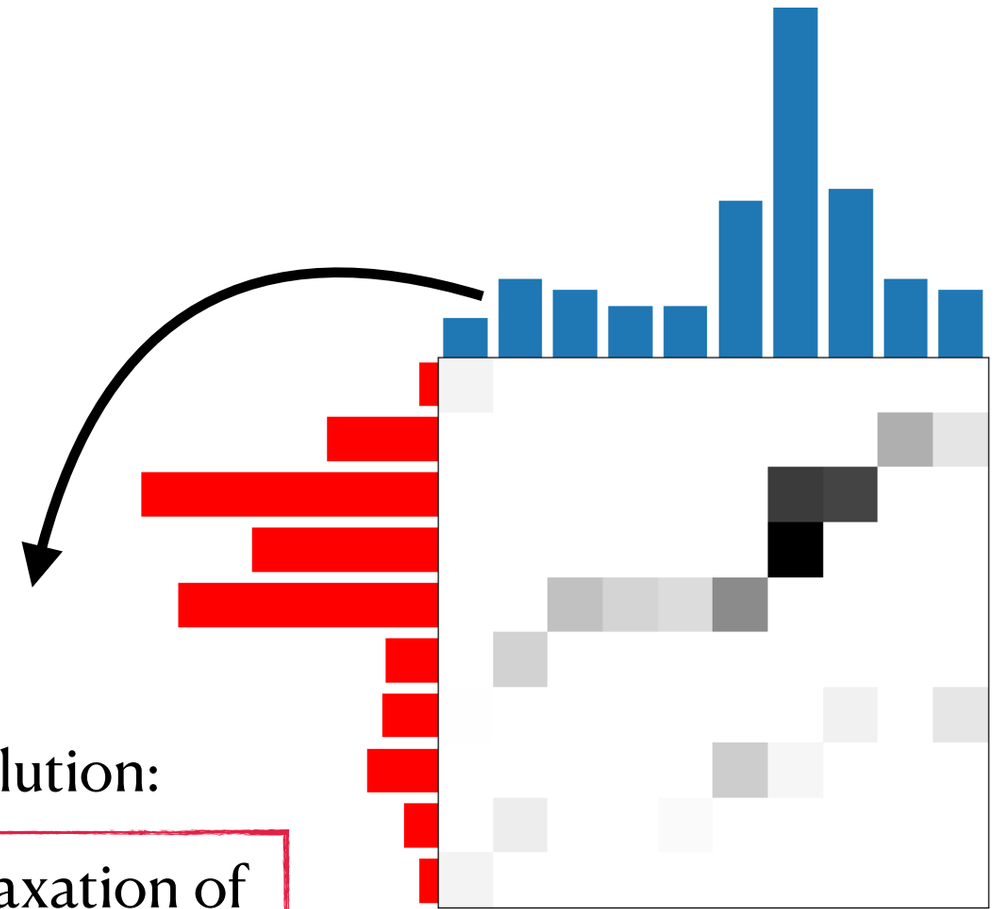
https://arxiv.org/abs/1803.00567v4



Problem of Monge : « Mémoire sur la théorie des déblais et des remblais », 1776

One solution:

With relaxation of Kantorovich



Optimal Transport for distributions

- **Optimal Transport:** Consider two finite sets $\mathbb{X} = \{\mathbf{x}_i\}_{i=1}^{|\mathbb{X}|} \in \mathbb{R}^{q \times |\mathbb{X}|}$ and $\mathbb{X}'$ and two distributions on these

$$\mu = \sum_{\mathbf{x}_i \in \mathbb{X}} a_i \delta_{\mathbf{x}_i} \quad \text{and} \quad \nu = \sum_{\mathbf{x}'_i \in \mathbb{X}'} b_i \delta_{\mathbf{x}'_i} \text{ with } a_i \geq 0,\ b_i \geq 0 \text{ and } \sum_{i=1}^{n} a_i = 1, \sum_{i=1}^{n'} b_i = 1$$

- Given a cost function $c : \mathbb{R}^q \times \mathbb{R}^q \to \mathbb{R}_+$, one builds the **2-Wasserstein distance** $\mathscr{W}_2$ as:

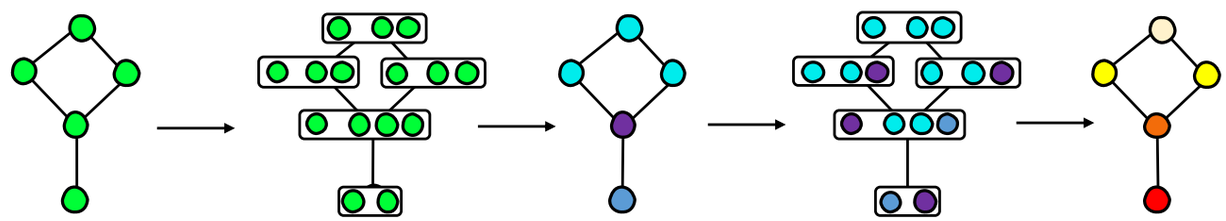
$$\mathscr{W}_2(\mu, \nu) = \inf_{\pi_{i,j} \in \Pi_{a,b}} \left(\sum_{i,j=1}^{n,n'} \pi_{i,j} c(\mathbf{x}_i, \mathbf{x}'_j)^2 \right)^{\frac{1}{2}}$$

where $\Pi_{a,b}$ is the set of joint distributions on $\mathbb{X} \times \mathbb{X}'$

whose marginals are the distributions $\mu = \sum_{\mathbf{x}'_i \in \mathbb{X}'} \pi(\cdot, \mathbf{x}'_i)$ and $\nu = \sum_{\mathbf{x}_i \in \mathbb{X}} \pi(\mathbf{x}_i, \cdot)$

Optimal Transport for Graphs

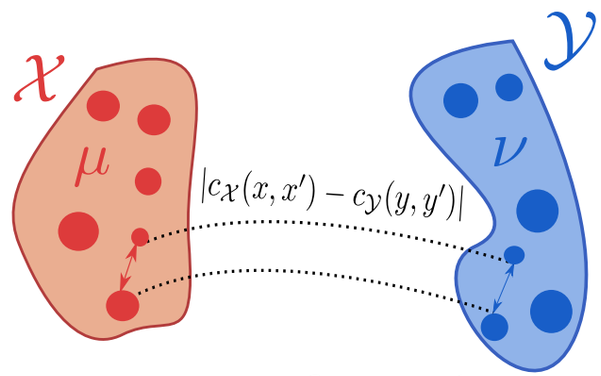
- For Graphs: one has to **Associate a distribution to a graph**
  - A first solution: rely on the the Weisfeiler-Lehman test
  - cf. [Togninalli et al., "Wasserstein Weisfeiler-Lehman graph kernels" NeurIPS 2019]



- A 2nd solution: **Comparison through probabilistic models of graph signals**
  - ["Graph Optimal Transport", H. Maretic et al. NeuRIPS 2019]
  - for a graph $\mathcal{G}$ with Laplacian L, one considers: $x \sim \nu^{\mathcal{G}} = \mathcal{N}(0, L^{\dagger})$
  - then: compute the 2-Wasserstein distance between Gaussian signals
  - allows graph alignment, gives a structurally-meaningful graph distance,...

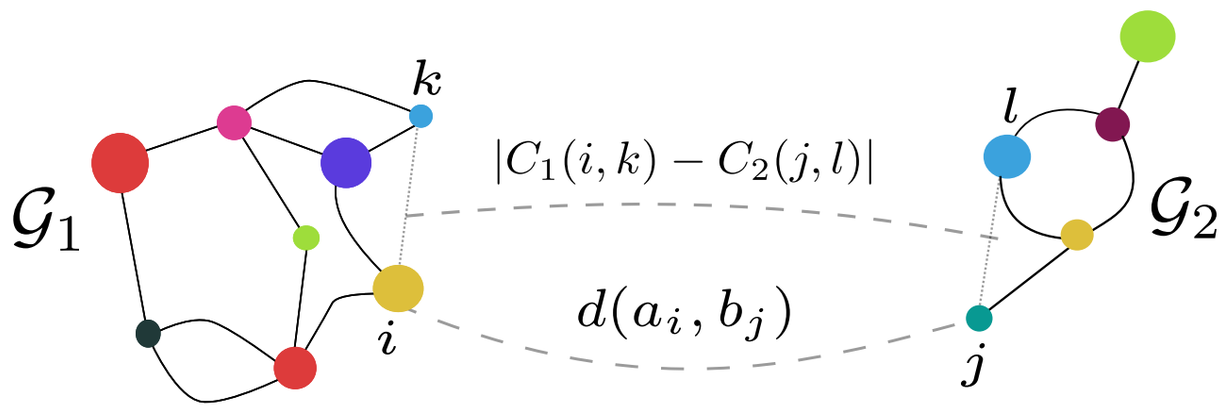
Optimal Transport for ~~Graphs or~~ Attributed Graphs

- A third solution: **The Gromov-Wasserstein distance**
  - Mémoli, Found. Comp. Math. 2011; Peyré, Cuturi, Solomon, ICML 2016
  - structures are compared through their pairwise distances
  - cf. also N. Courty, R. Flamary, T. Vayer [PhD 2020]



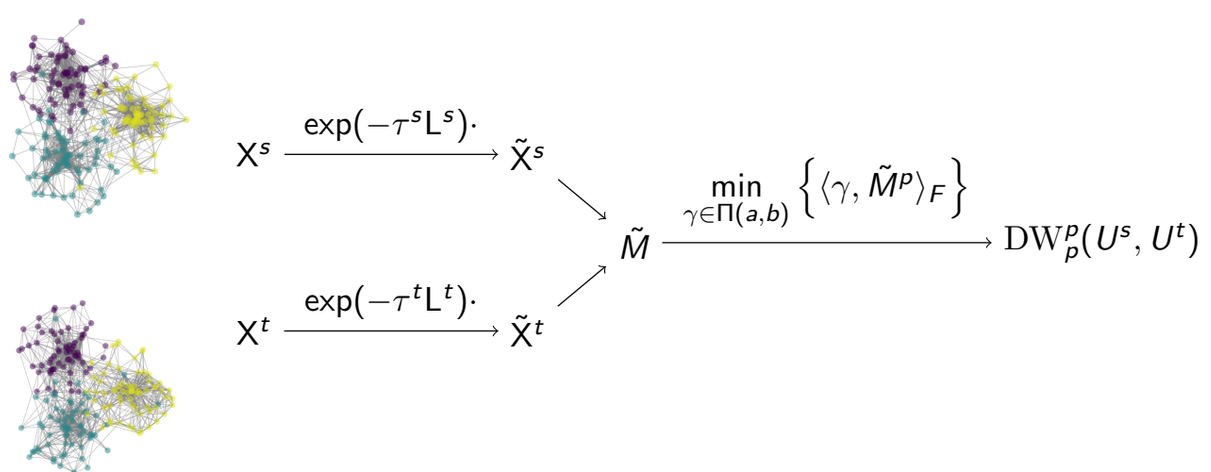
- One can then **combine Attributes and Gromov-Wasserstein** characterisation of graphs

  "Fused Gromov-Wasserstein distance" [Vayer et al., ICML 2019]



Optimal Transport and Graph Signal Processing for Attributed Graphs

- We can leverage (**combine**) that: **Optimal Transport ; Diffusion distance ; and Graph Signal Processing** (i.e., process signals by L)
- We generalize the previous ideas, and we consider:
  - two graphs of sizes n and m and their associated Laplacians: $\mathbf{L}^s$ and $\mathbf{L}^t$
  - the features of these *source* and *target* graphs: $\mathbf{X} \in \mathbb{R}^{m \times r}; \mathbf{Y} \in \mathbb{R}^{n \times r}$
  - a cost function between features: $M(\mathbf{X}, \mathbf{Y}) = [d(x_i, y_j)]$ for any $\mathbf{X} \in \mathbb{R}^{m \times r}; \mathbf{Y} \in \mathbb{R}^{n \times r}$
  - the diffused features: $\tilde{\mathbf{X}} = \exp(-\tau^s \mathbf{L}^s) \cdot \mathbf{X}$ and $\tilde{\mathbf{Y}} = \exp(-\tau^t \mathbf{L}^t) \cdot \mathbf{Y}$

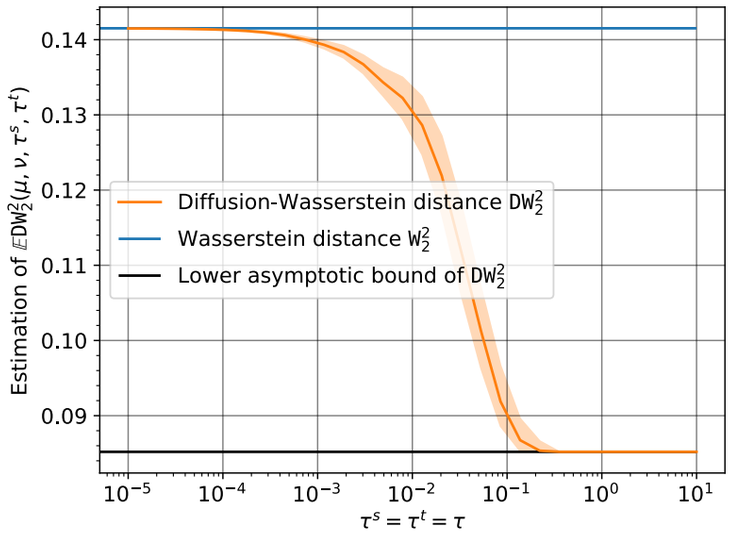


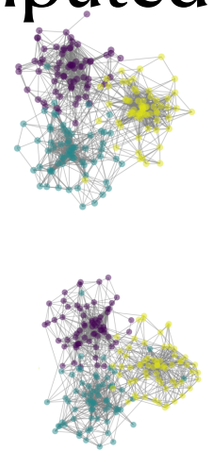
The Diffusion Wasserstein Distances for Attributed Graphs

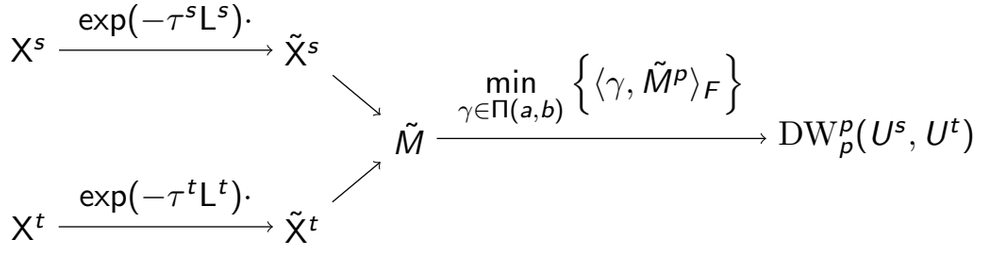
- Then, we define it as:

$$\mathtt{DW}_p^p(\mu, \nu \mid \tau^s, \tau^t) = \min_{\gamma \in \Pi(a,b)} \langle \gamma, \tilde{M}^p \rangle.$$

- **Theoretically**, it has good properties:
  - it is a distance
  - we have bounds for small and large τ
  - it's efficient to be computed, more than Fused GW

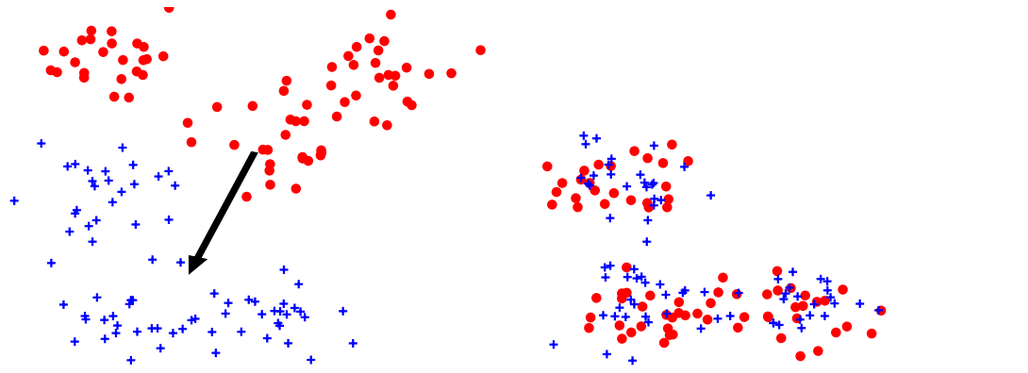






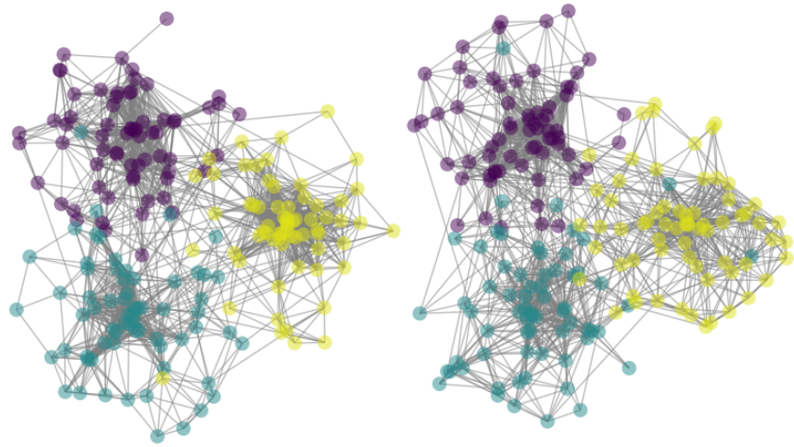
The Diffusion Wasserstein Distances for Attributed Graphs

$$\mathtt{DW}_p^p(\mu, \nu \mid \tau^s, \tau^t) = \min_{\gamma \in \Pi(a,b)} \langle \gamma, \tilde{M}^p \rangle.$$

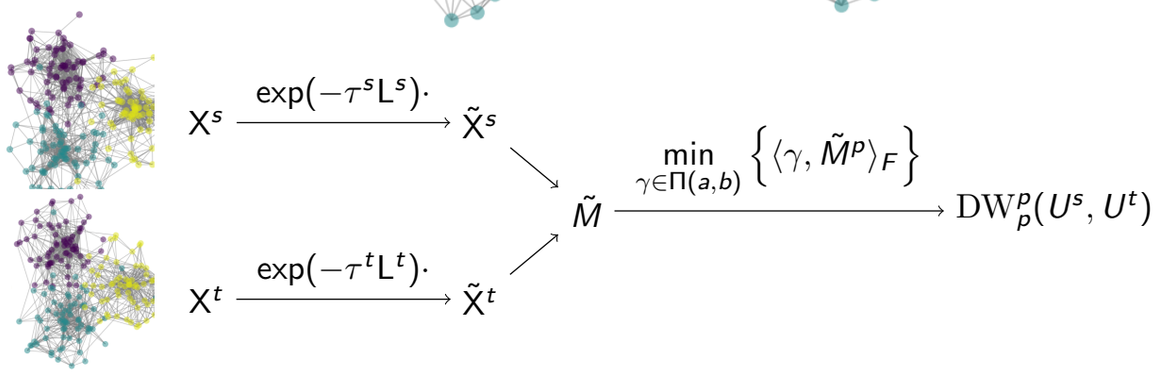


(a) Distributions before alignment. (b) Distributions after alignment.

- **Experimentally**, it works well: the task for comparison is **Domain Adaptation**
  - by itself a cheap way for **DA** on Attr. Graphs



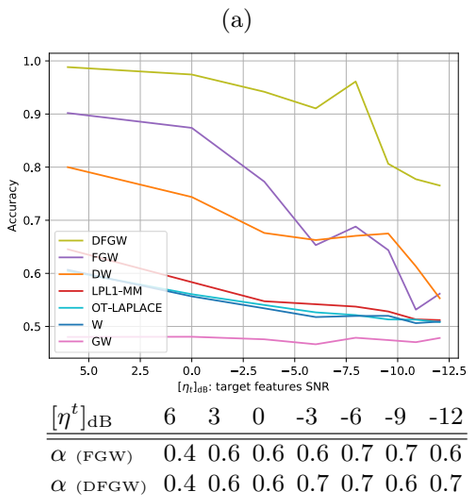
  - can be combined with Fused GW, for an even better *DifFused GW* distance, which has best perf. !



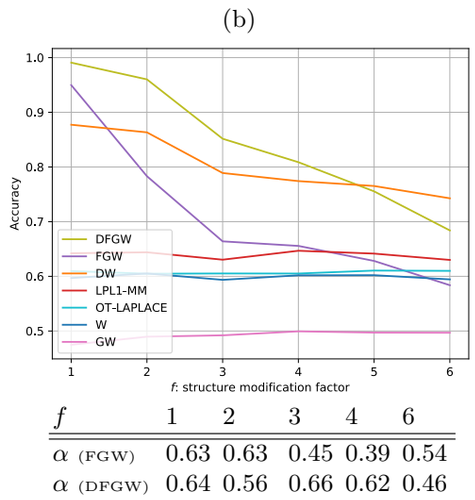
The Diffusion Wasserstein Distances for Attributed Graphs

$$\mathtt{DW}_p^p(\mu,\nu \mid \tau^s,\tau^t) = \min_{\gamma\in\Pi(a,b)} \langle \gamma, \tilde{M}^p \rangle.$$

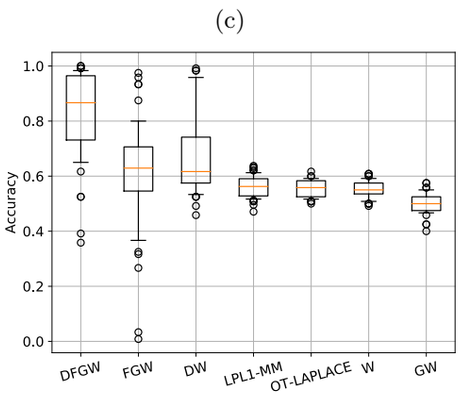
- **Experimentally**, it works well: the task for comparison is **Domain Adaptation**

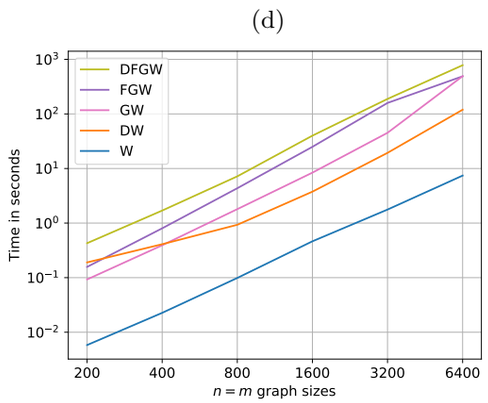


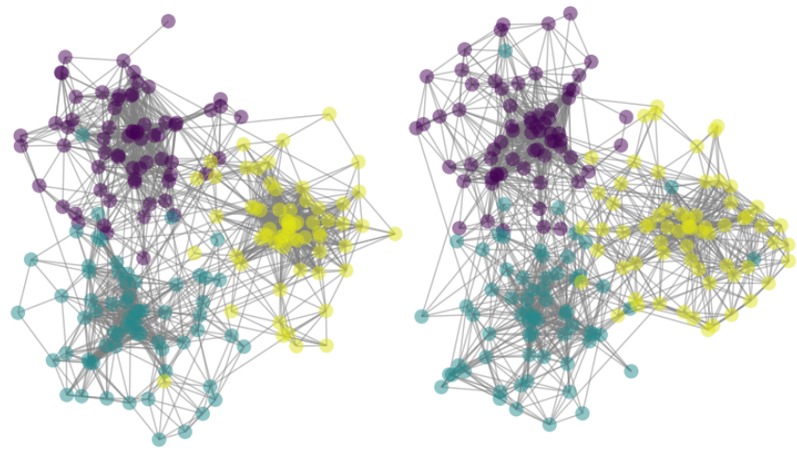
$[\eta^t]_{\mathrm{dB}}$	6	3	0	-3	-6	-9	-12
α (FGW)	0.4	0.6	0.6	0.6	0.7	0.7	0.6
α (DFGW)	0.4	0.6	0.6	0.7	0.7	0.6	0.7

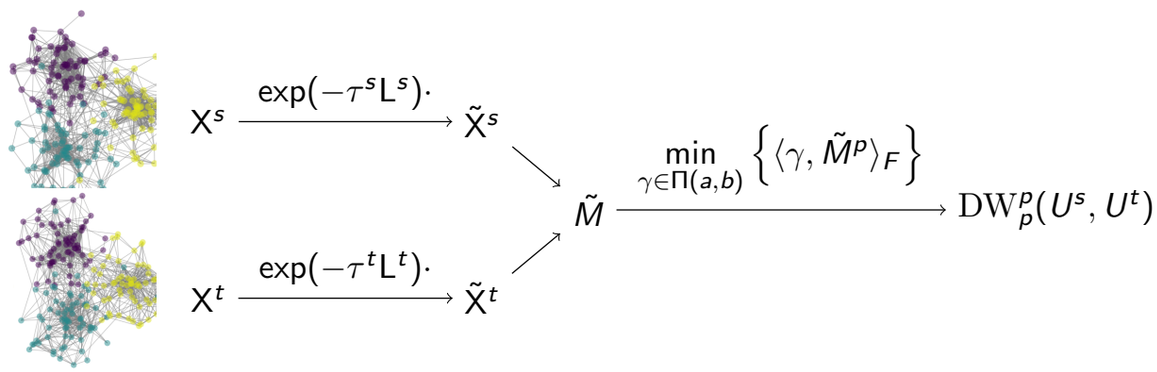


f	1	2	3	4	6
α (FGW)	0.63	0.63	0.45	0.39	0.54
α (DFGW)	0.64	0.56	0.66	0.62	0.46









from [Barbe et al., ECML-PKDD 2020]

(1) Filters on graphs - sequel

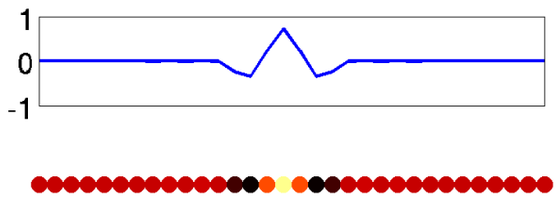
- On board =
  - Various definitions of filters in graphs
  - Implementation of graph filters
  - Shift operators and frequencies

(2) Filters on graphs => multiresolution

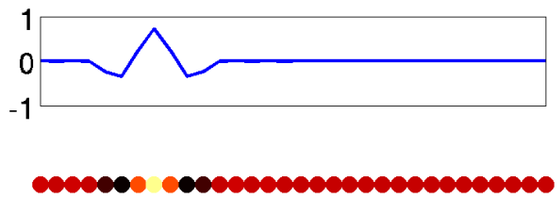
Example 2: Wavelets for graph signals

- Wavelet = a local function, acting as filter around a chosen scale, defined scaling and dilating

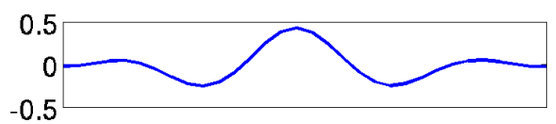
  A wavelet:



– Translated:

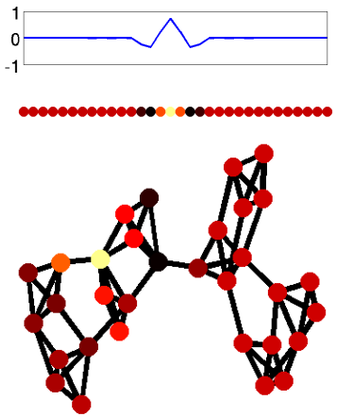


– Scaled

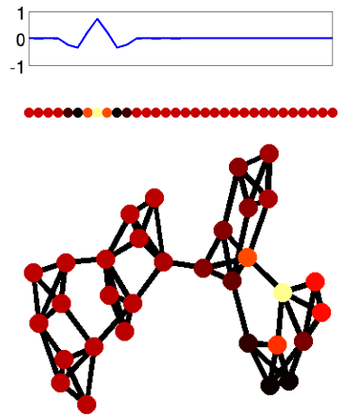


- on Graphs ?

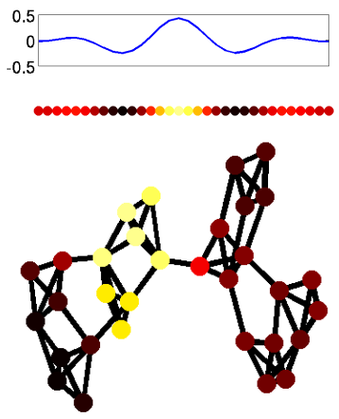
A WAVELET:



TRANSLATING:



SCALING:



(2) Filters on graphs => multiresolution

Classical wavelets $\xrightarrow{\textit{by analogy}}$ Graph wavelets

[Hammond et al., ACHA 2011]

The wavelet at scale s centered around a is given by:

$$\psi_{s,u}(t) = \frac{1}{s}\psi\left(\frac{t-u}{s}\right) = \int_{-\infty}^{\infty} \hat{\delta}_u(\omega)\hat{\psi}(s\omega)\exp^{i\omega t}\, d\omega$$

	Classical (continuous) world	Graph world
Real domain	t	node a
Fourier domain	ω	eigenvalues λ_i
Filter kernel	$\hat{\psi}(\omega)$	$h(\lambda_i) \Leftrightarrow \hat{\boldsymbol{H}}$
Filter bank	$\hat{\psi}(s\omega)$	$h(s\lambda_i) \Leftrightarrow \hat{\boldsymbol{H}}_{\boldsymbol{s}}$
Fourier modes	$\exp^{-i\omega t}$	eigenvectors χ_i
Fourier transf. of x	$\hat{x}(\omega) = \int_{-\infty}^{\infty} x(t)\exp^{-i\omega t}\, dt$	$\hat{x} = \boldsymbol{\chi}^\top x$

In the graph world by analogy: $\boxed{\psi_{s,a} = \boldsymbol{\chi}\,\hat{\boldsymbol{H}}_{\boldsymbol{s}}\hat{\delta}_a = \boldsymbol{\chi}\,\hat{\boldsymbol{H}}_{\boldsymbol{s}}\,\boldsymbol{\chi}^\top\,\delta_a}$

(2) Filters on graphs => multiresolution

Generalized translations

[Shuman, Ricaud, Vandergheynst, 2014]

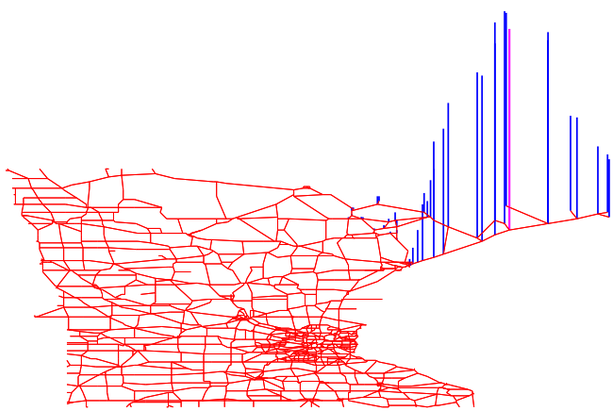
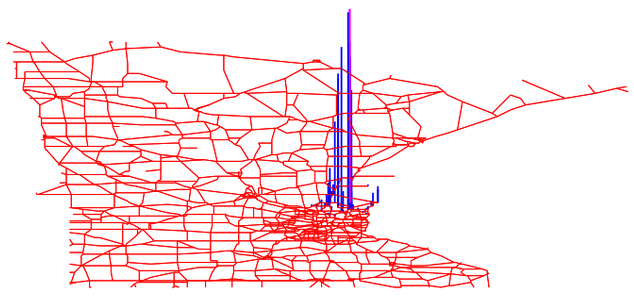
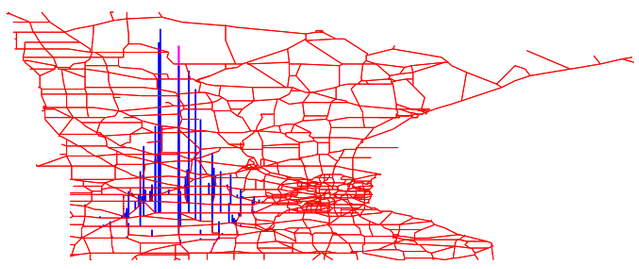
- Classical translation (continuous world)

$$(T_\tau g)(t) = g(t-\tau) = \int_{\mathbb{R}} \hat{g}(\xi) e^{-i2\pi\tau\xi} e^{-i2\pi t\xi} d\xi$$

- Graph translations by fundamental analogy:

$$(T_n f)(a) = \sum_{i=0}^{N-1} \hat{f}(i) \chi_i^*(n) \chi_i(a)$$

- Example on the Minnesota road networks

(2) Filters on graphs => multiresolution

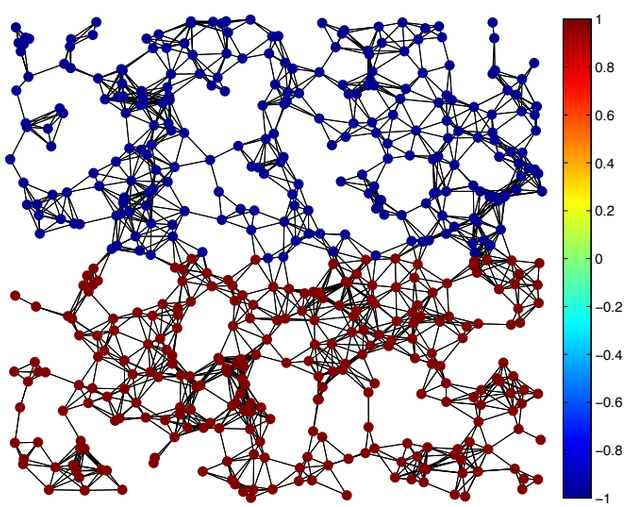
: Recovery of signals on graphs

- Denoising of a signal with Wavelet regularization, *W* being the direct wavelet transform and *c* the wavelet coefficients of the graph signal:

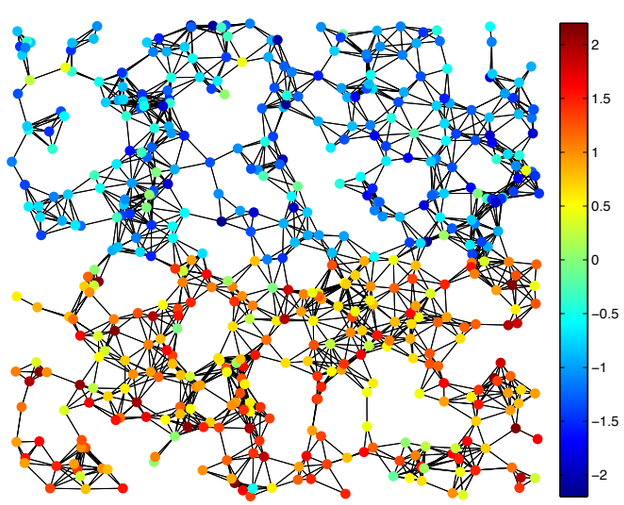
$$\arg\min_{c} ||W^{\top} c - y||_2^2 + \gamma ||c||_1$$

- Solution with IST algorithm with a step $\tau < 2/||W^{\top}||^2$:

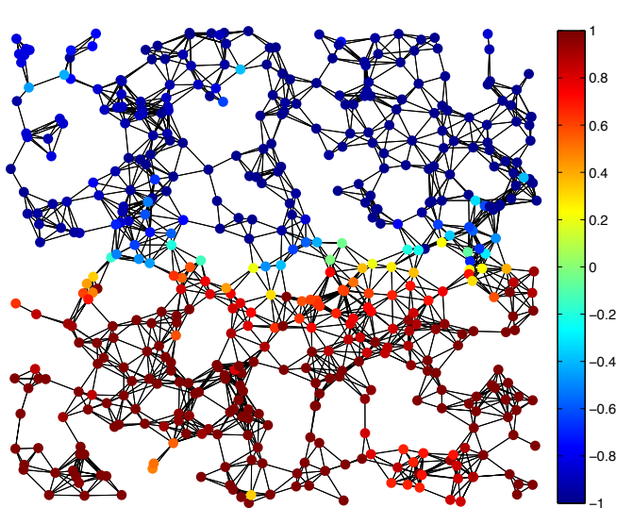
$$c^{(k)} = S_{\tau}\left(c^{(k-1)} + \tau W(y - W^{\top} c^{(k-1)})\right)$$



Original



Noisy



Denoised

(2) Filters on graphs => multiresolution

GraphWave (2018)

Learning Structural Node Embeddings via Diffusion Wavelets

Claire Donnat, Marinka Zitnik, David Hallac, Jure Leskovec
Stanford University
{cdonnat,marinka,hallac,jure}@stanford.edu

KDD '18, August 19–23, 2018, London, United Kingdom

- Use wavelets to have a multi-scale view of the neighbourhood of each node

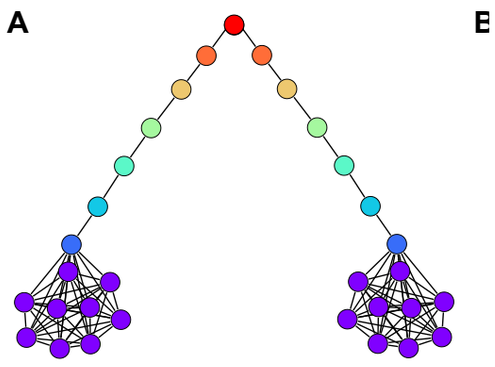
$$\Psi_a = U\,\mathrm{Diag}(g_s(\lambda_1), \ldots, g_s(\lambda_N))U^T\delta_a, \qquad (1)$$

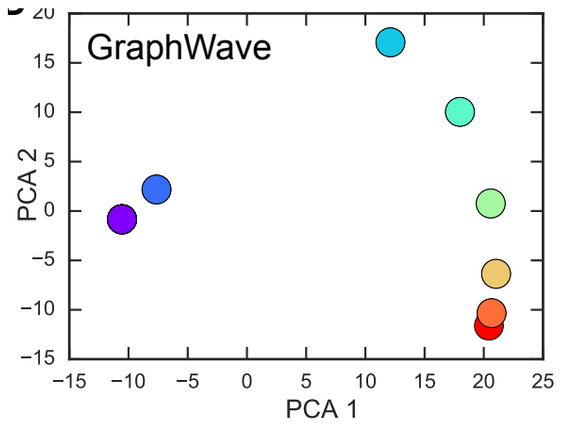
- Then embed each node with the wavelet coefficients

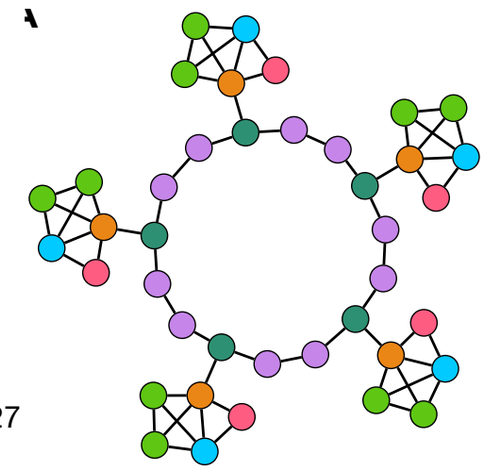
Algorithm 1 Learning structural embeddings in GraphWave.

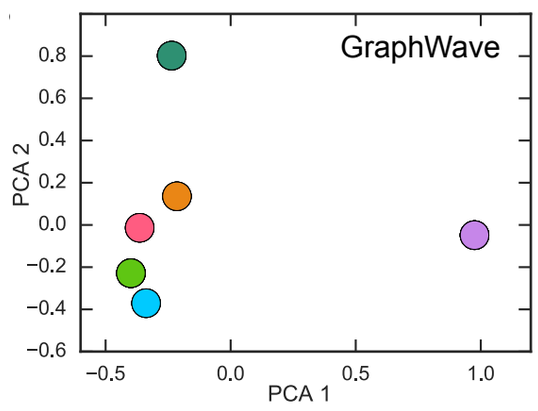
Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, scale s, evenly spaced sampling points $\{t_1, t_2, \ldots, t_d\}$
Output: Structural embedding $\chi_a \in \mathbb{R}^{2d}$ for every node $a \in \mathcal{V}$
Compute $\Psi = Ug_s(\Lambda)U^T$ (Eq. (1))
for $t \in \{t_1, t_2, \ldots, t_d\}$ **do**
    Compute $\phi(t) =$ column-wise mean$(e^{it\Psi}) \in \mathbb{R}^N$
    for $a \in \mathcal{V}$ **do**
        Append $\mathrm{Re}(\phi_a(t))$ and $\mathrm{Im}(\phi_a(t))$ to χ_a

- Finally: cluster or classify



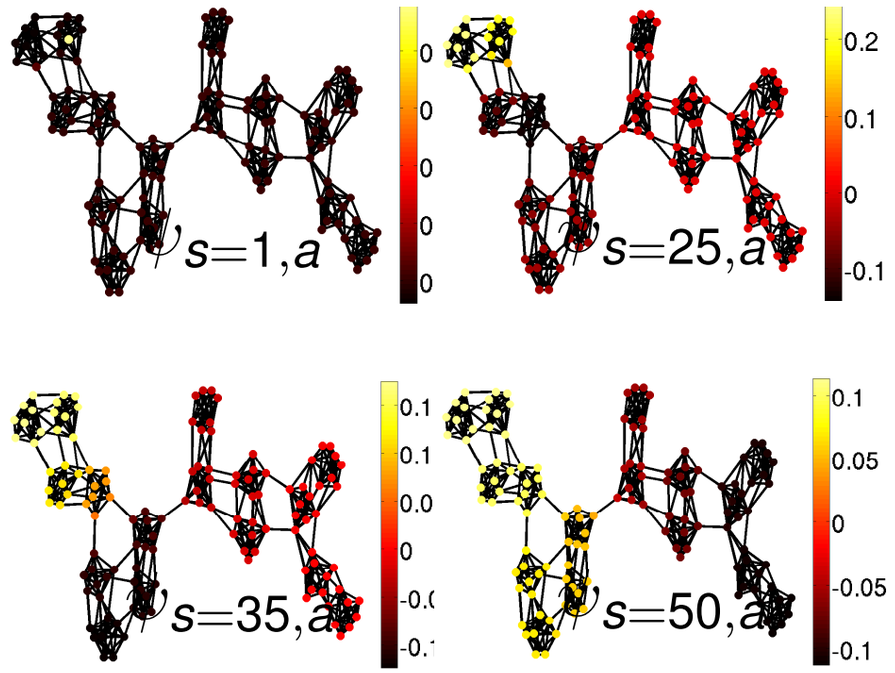






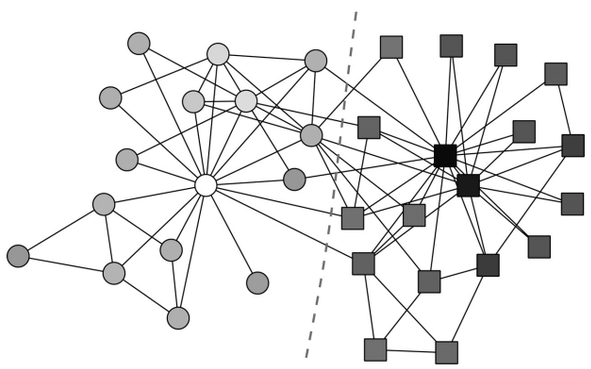
(2) Filters on graphs => multiresolution

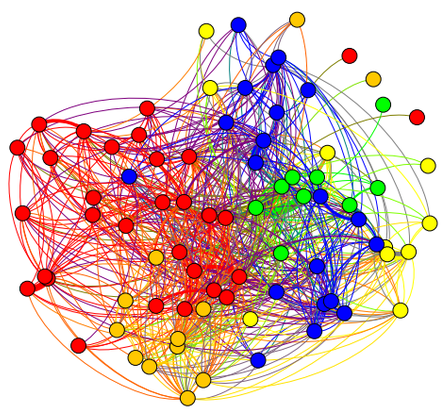
Find multiscale communities in complex: with wavelets on graphs

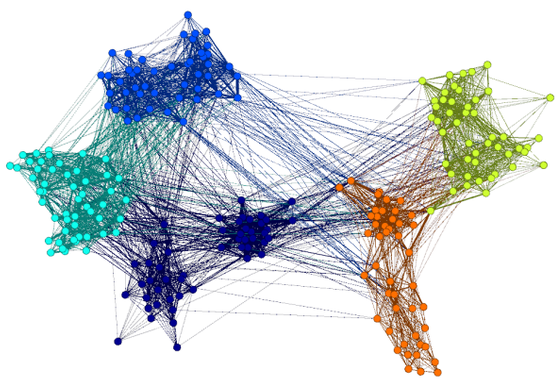


- A means to find communities in networks ? Yes
- Some examples of social networks:

Zachary Karatee Club; Sociopatterns data (ISI Turin, CPT Marseille)



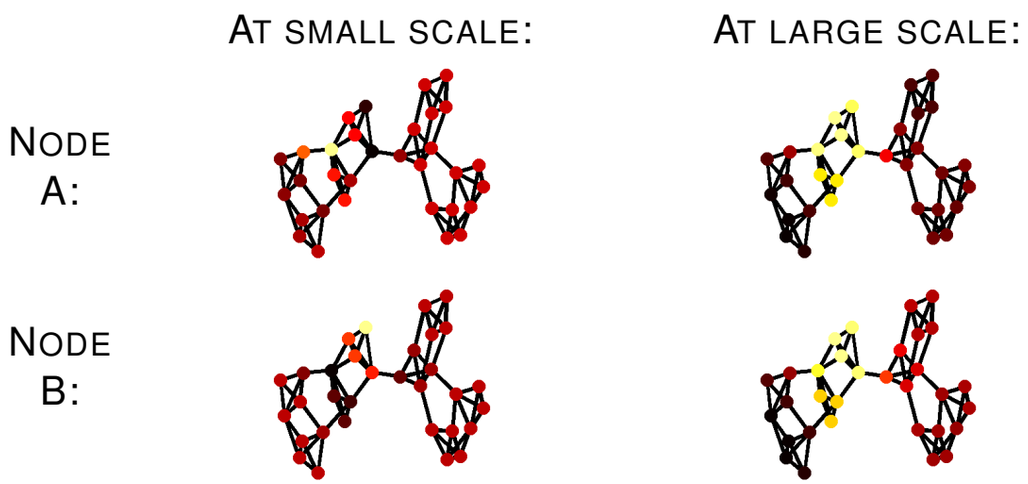




(2) Filters on graphs => multiresolution

Find multiscale communities in complex : with wavelets on graphs

Filterbanks-based spectral graph clustering



- Similarity: $\boldsymbol{D}_s(a,b) = 1 - \frac{\boldsymbol{f}_{s,a}^\top \boldsymbol{f}_{s,b}}{||\boldsymbol{f}_{s,a}||_2 \; ||\boldsymbol{f}_{s,b}||_2}$.
- Classification using hierarchical agglomerative clustering with average-linkage
- (Not detailed): Add stochasticity in the measurement

$$\mathbf{f}_{s,a} = \boldsymbol{r}^\top \mathbf{H}_s \boldsymbol{\chi}^{-1} \delta_a$$

where $\boldsymbol{r} \in \mathbb{R}^{N \times \eta}$ is i.i.d., centered, normal

- N. Tremblay and P. Borgnat, Graph Wavelets for Multiscale Community Mining, IEEE TSP, 62: 20, p. 5227, 2014

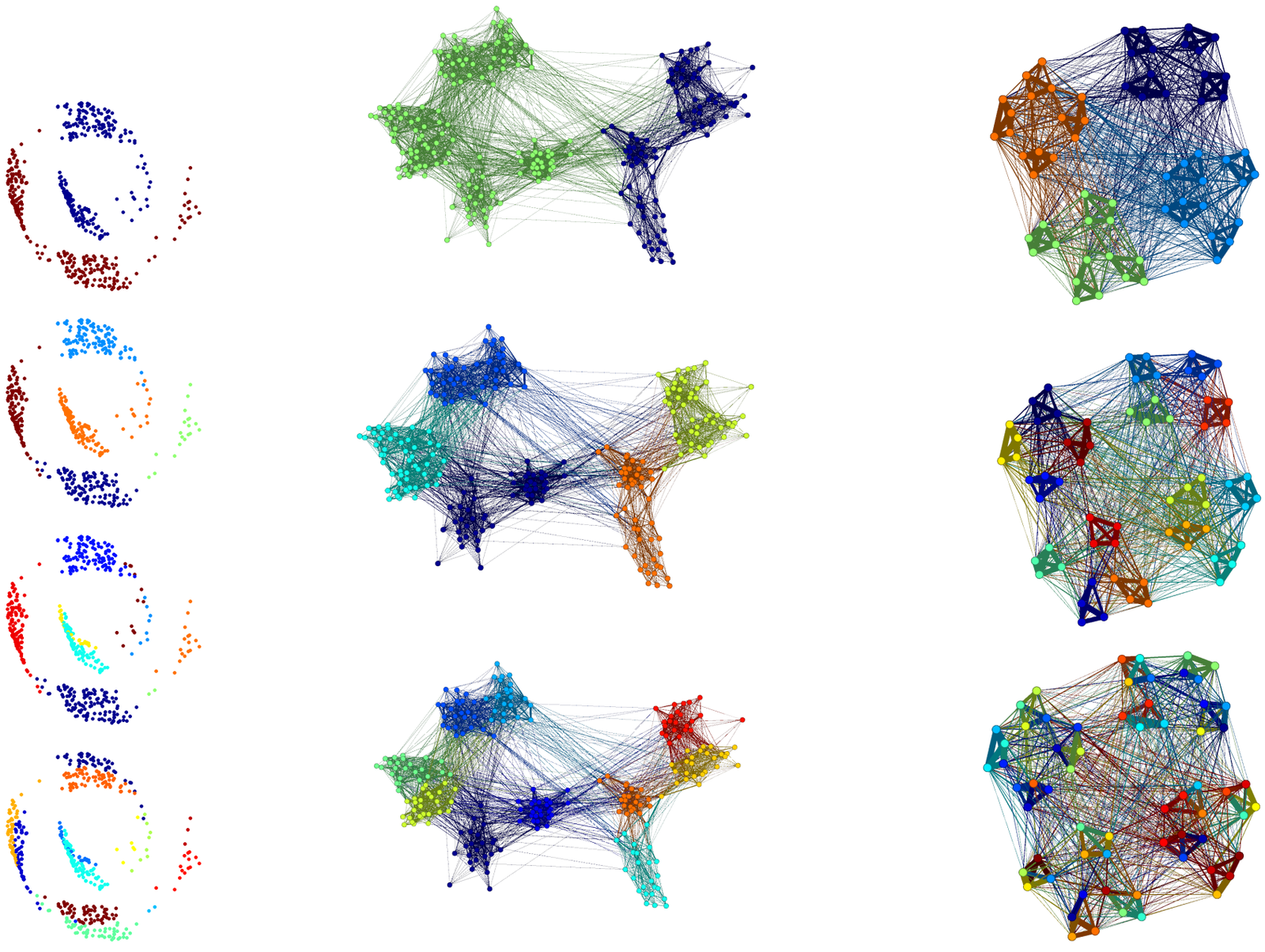
- N. Tremblay, G. Puy, P. Borgnat, R. Gribonval, P. Vandergheynst, ICASSP 2016

(2) Filters on graphs => multiresolution

Find communities in complex : with wavelets on graphs

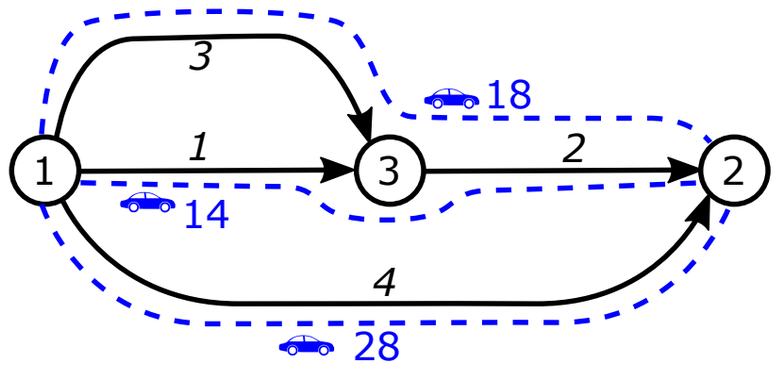
Multiscale community detection on networks

[Tremblay, Borgnat 2014]



— A break just for fun: more complex inverse problems on graphs —

An Estimation problem [Michau, 2017]
Link Dependent Origin-Destination Matrix



- LODM estimation engineered as an inverse problem

$$\underline{\underline{\widehat{Q}}} \in \underset{\underline{\underline{Q}}}{\text{Argmin}} \left\{ \gamma_{TC} f_{TC}(\underline{\underline{Q}}) + \gamma_P f_P(\underline{\underline{Q}}) + \gamma_C f_C(\underline{\underline{Q}}) + \gamma_K f_K(\underline{\underline{Q}}) + \gamma_{TV} f_{TV}(\underline{\underline{Q}}) \right\}$$

- Prior information available:
  - $\underline{\underline{B}}$ trajectories that give sampled LOD counts
  - $\underline{q}$ counts on roads (without OD information)

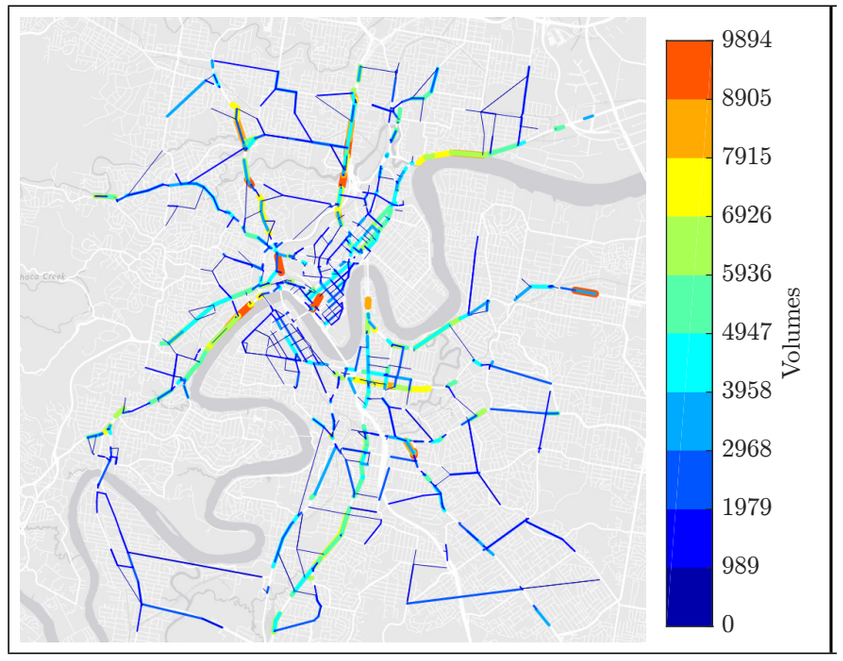
- Assumption of small Total Variation:

$$f_{TV}(\underline{\underline{Q}}) = \sum_{i \sim \mathcal{N}_{i'}} \sum_{j,l} \omega_{ii'} |Q^l_{ij} - Q^l_{i'j}| + \sum_{j \sim \mathcal{N}_{j'}} \sum_{i,l} \omega_{jj'} |Q^l_{ij} - Q^l_{ij'}|$$

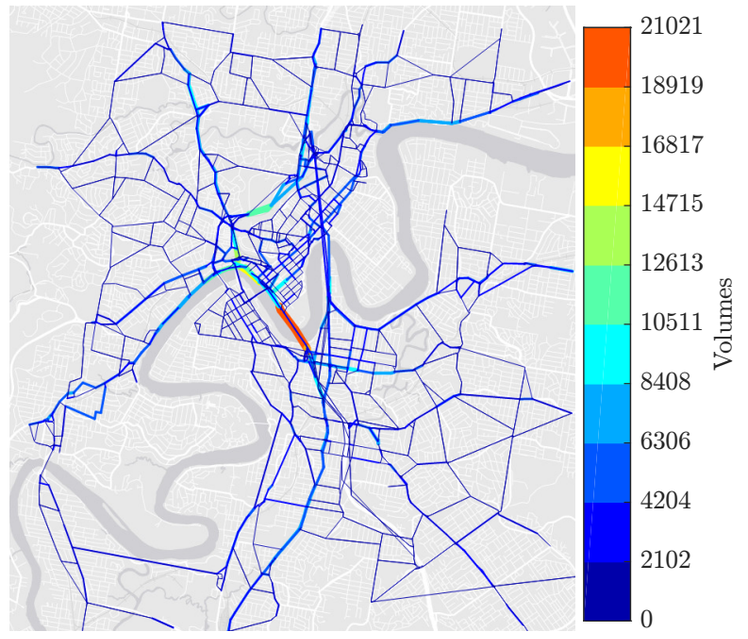
where $\mathcal{N}_{i'}$ is the neighbourhood of i' and $\omega_{ii'} \geq 0$ are weights (e.g., taken as exp(-distance/d_0))

The Brisbane case study – results

- Comparison of traffic counts on roads:



Measured (on 36%)



Estimated here; more smooth!

Create **graphs which describe data**
from classical methods,
or statistical models,
or considerations from GSP

Create a graph to represent the data

Objective: capture similarities between data points

- This is a standard step in classification / clustering!
- Hence, several manners to code these similarities in a graph:

selecting k-nearest neighbours of each point with distance $d(x_i, x_j)$

OR

selecting all points in a neighbourhood $d(x_i, x_j) \leqslant \epsilon$

Create a graph to represent the data

Objective: capture similarities between data points

Distance functions

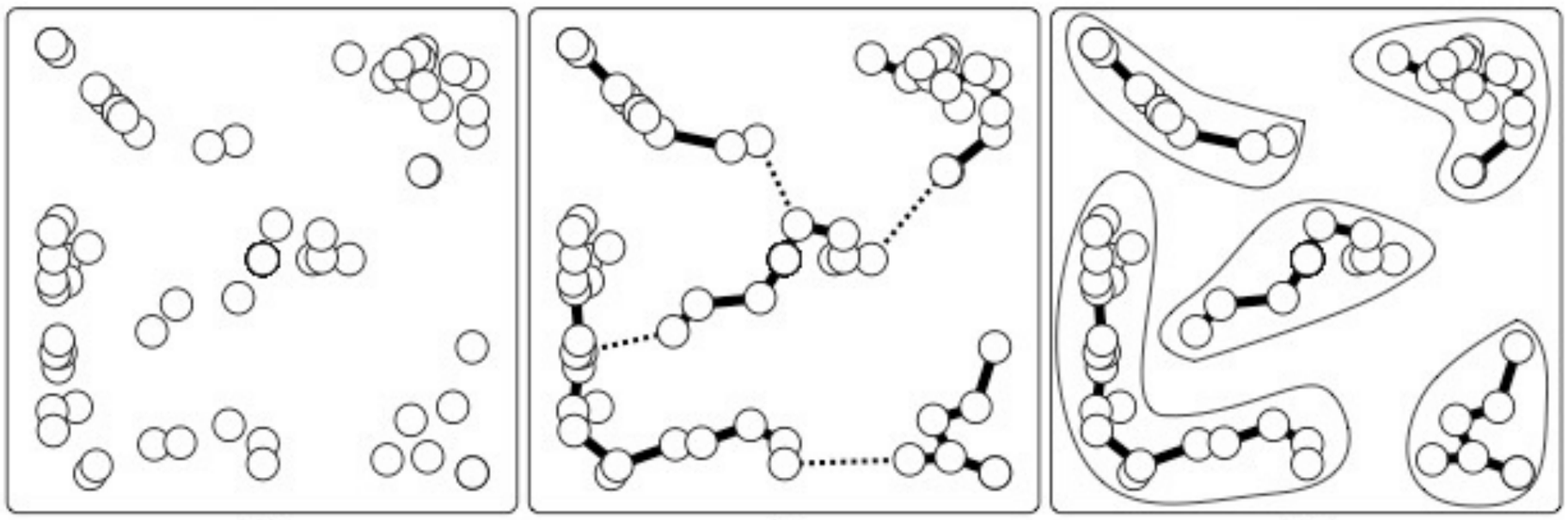
- Given X_u and X_v, how far are they from one another ?
- Euclidean distance (or its square): $\sum_n (x_{nu} - x_{nv})^2$
- ℓ_1 or Manhattan distance: $\sum_n |x_{nu} - x_{nv}|$
- Mahalanobis distance: $\sqrt{\sum_n (x_{nu} - x_{nv})^2 / \sigma_n^2}$ or more generally $\sqrt{(X_u - X_v)^\top \mathbf{C}^{-1} (X_u - X_v)}$
- From correlations, e.g. $1 - X_u \cdot X_v$
- From kernels: $K(X_u, X_v)$, with K a "kernel"
  eg. Gaussian one: $\exp(-(X_u - X_v)^2 / 2\sigma^2)$
- ...

Create a graph to represent the data

Objective: keep strong similarities (only) between data points

Great a graph "connecting the dots", i.e. find edges
to connect data points.
Several possibilities:

- **Mininimal Spanning Tree:** the tree with smallest sum of edge lengths connecting all nodes

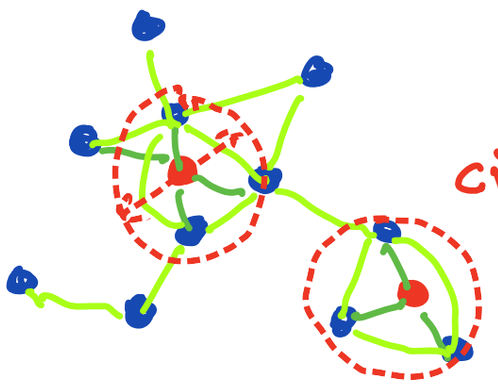


Create a graph to represent the data

Objective: keep strong similarities (only) between data points

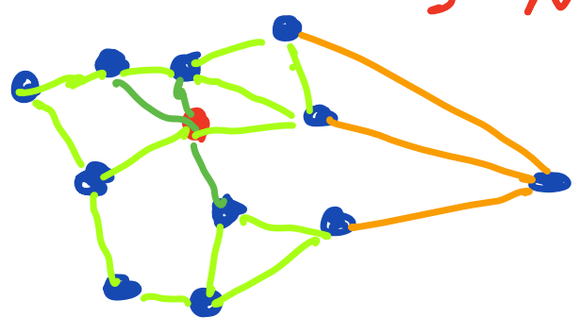
Great a graph "connecting the dots", i.e. find edges
to connect data points.
Several possibilities:

The ε-neighborhood graph: $d(x_i, x_j) \leqslant \epsilon$ (1)



circles of radius ε

3-NN graph



⚠ with outliers

Create a graph to represent the data

Objective: keep strong similarities (only) between data points

Great a graph "connecting the dots", i.e. find edges
to connect data points.
Several possibilities:

k-nearest neighbor graphs: with distance $d(x_i, x_j)$

3-NN graph

⚠ with outliers

Note : . $d \geq k$ if symmetrized
. Very sparse if kNN mutual neighbors

Create a graph to represent the data

Objective: keep strong similarities (only?) between data points

Great a graph "connecting the dots", i.e. find edges
to connect data points.
Several possibilities:

- **The fully connected graph:**

  connect all nodes with all other nodes, but with a weight on each

  edge, derived from some similarity function, going to 0 if distance goes to infinity

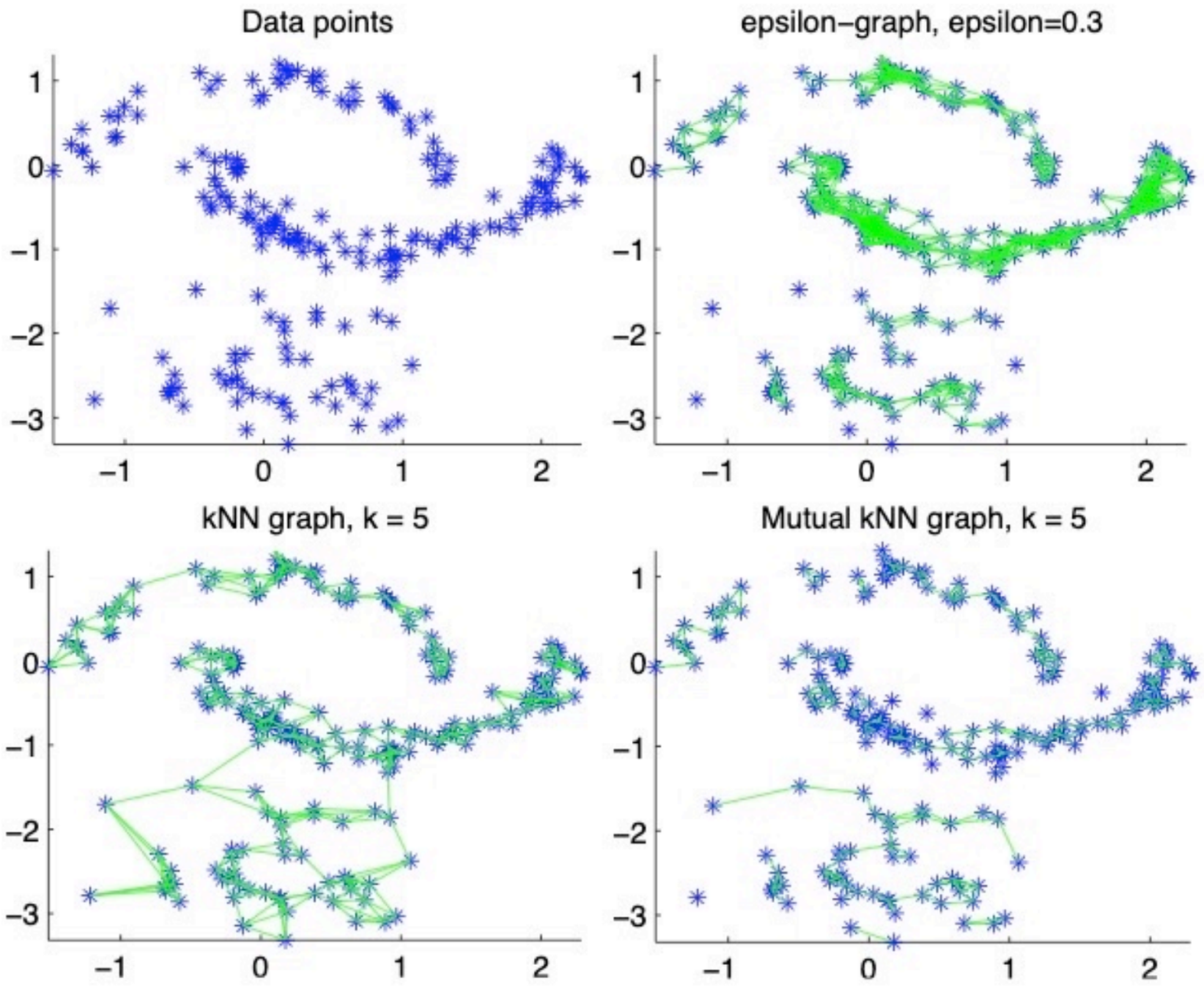
  Example: Gaussian similarity function $s(x_i, x_j) = \exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2})$

\ interest of the 3 previous solutions: **sparse graphs** !

for complete graph with similarity kernel: use **thresholding** to increase sparsity of the graph.

Create a graph to represent the data

Examples



A Tutorial on Spectral Clustering, Ulrike von Luxburg,

Create a graph to represent the data

Interlude: you know other methods!

2-a) model the local neighbourhood relationships between the data points

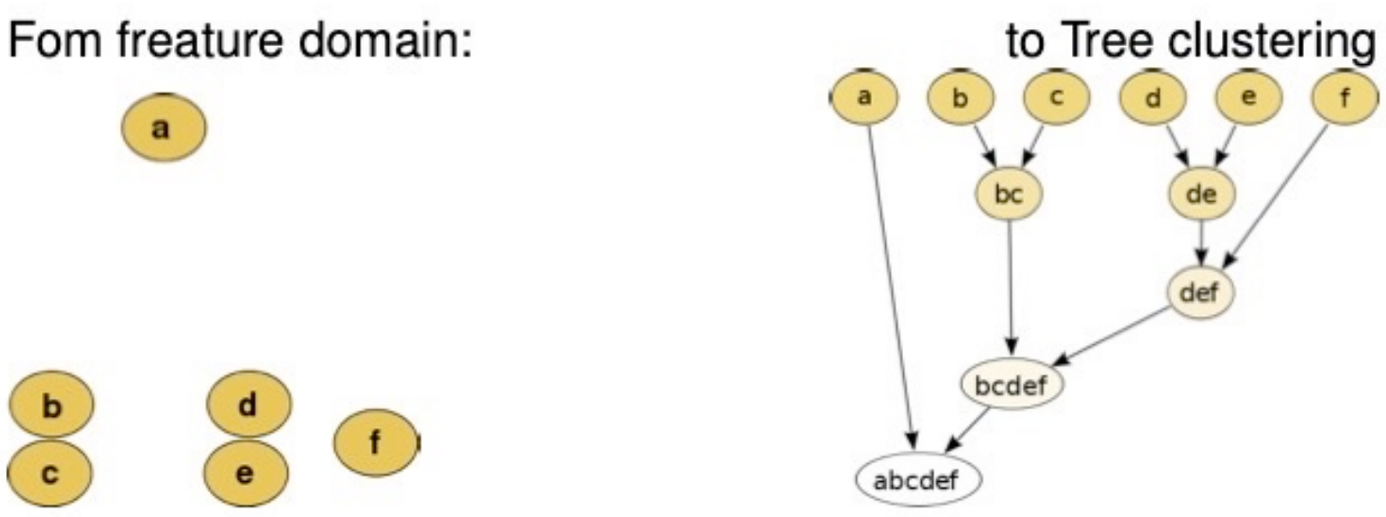
=> Just what we described

Create a graph to represent the data

Interlude: you know other methods!

2-b) Create a graph that clusters (or classifies) data points
A possible solution: **Hierarchical clustering**

- Main idea: group together closest points



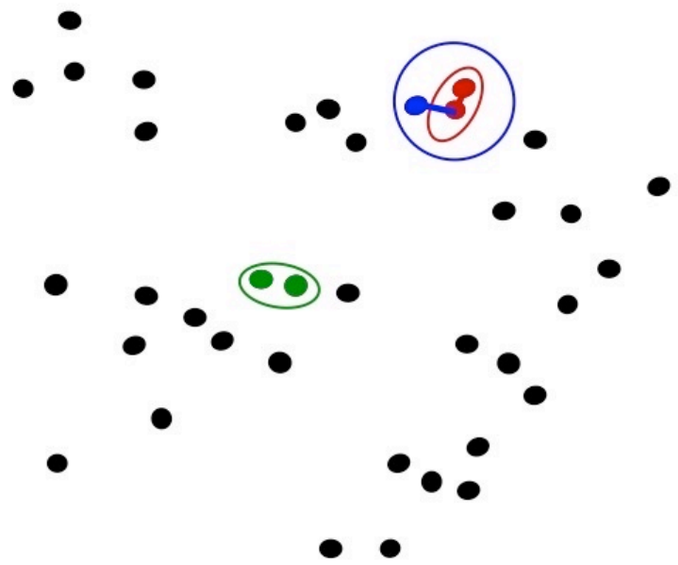
with two broad strategies: Agglomerative (a "bottom-up" approach) vs. Divisive (a "top-down" approach)

Create a graph to represent the data

Interlude: you know other methods!

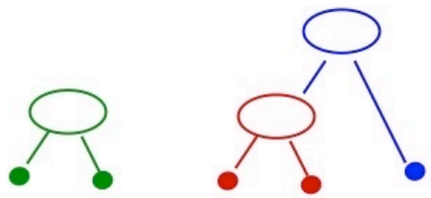
2-b) Create a graph that clusters (or classifies) data points
A possible solution: **Hierarchical clustering**

- Agglomerative clustering:
  - First merge very similar instances
  - Incrementally build larger clusters out of smaller clusters



- Algorithm:
  - Maintain a set of clusters
  - Initially, each instance in its own cluster
  - Repeat:
    - Pick the two closest clusters
    - Merge them into a new cluster
    - Stop when there's only one cluster left

- Produces not one clustering, but a family of clusterings represented by a dendrogram



Create a graph to represent the data

Interlude: you know other methods!

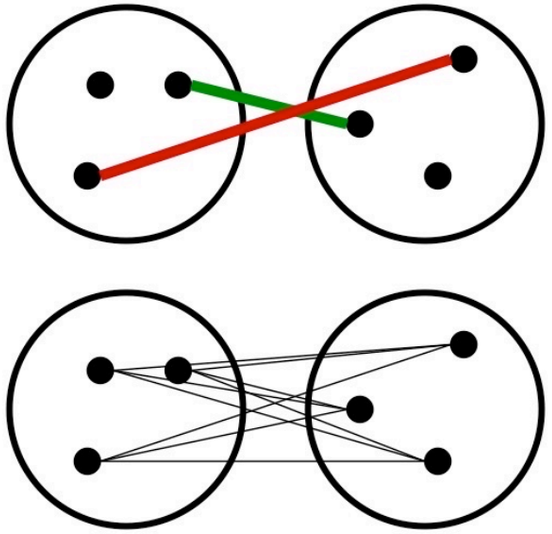
2-b) Create a graph that clusters (or classifies) data points
A possible solution: **Hierarchical clustering**

An issue involved in Agglomerative clustering

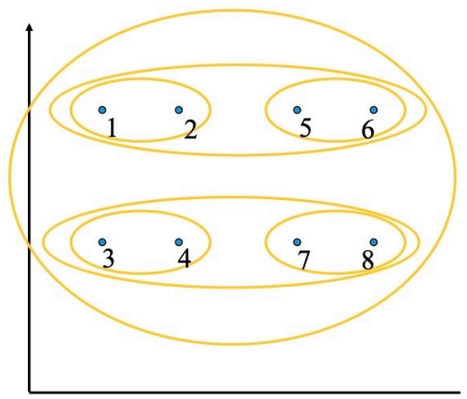
- How should we define “closest” for clusters with multiple elements?

- Many options:
  - Closest pair (single-link clustering)
  - Farthest pair (complete-link clustering)
  - Average of all pairs

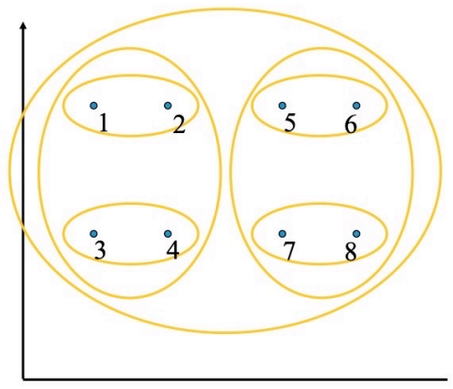
- Different choices create different clustering behaviors



Closest pair
(single-link clustering)



Farthest pair
(complete-link clustering)



[Pictures from Thorsten Joachims]

Create a graph to represent the data

Interlude: you know other methods!

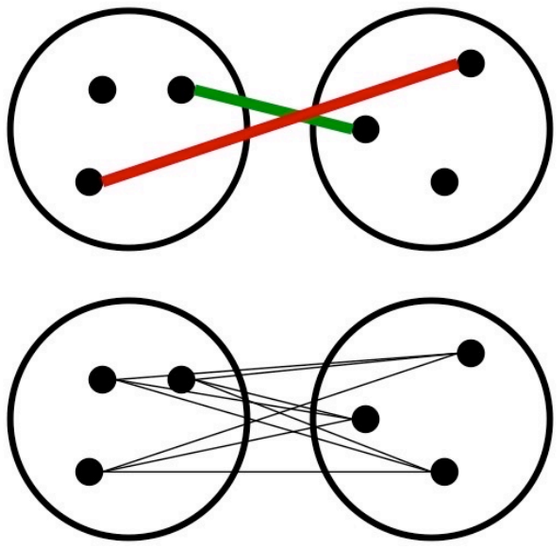
2-b) Create a graph that clusters (or classifies) data points
A possible solution: **Hierarchical clustering**

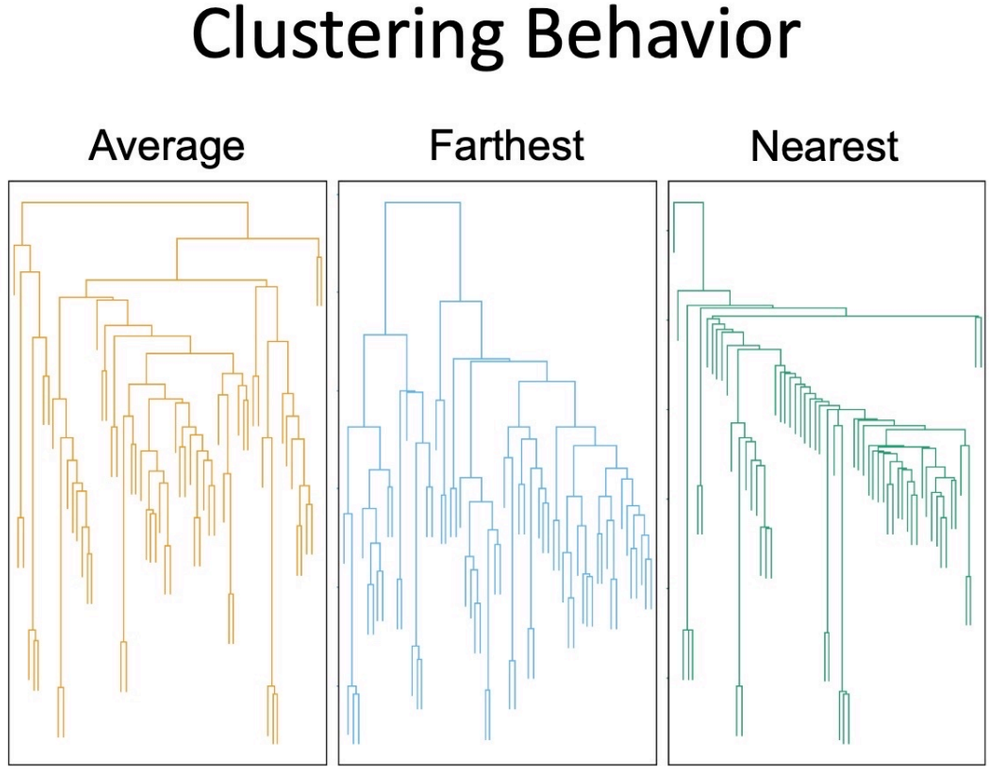
An issue involved in Agglomerative clustering

- How should we define "closest" for clusters with multiple elements?

- Many options:
  - Closest pair (single-link clustering)
  - Farthest pair (complete-link clustering)
  - Average of all pairs

- Different choices create different clustering behaviors





Mouse tumor data from [Hastie *et al.*]

Create a graph to represent the data

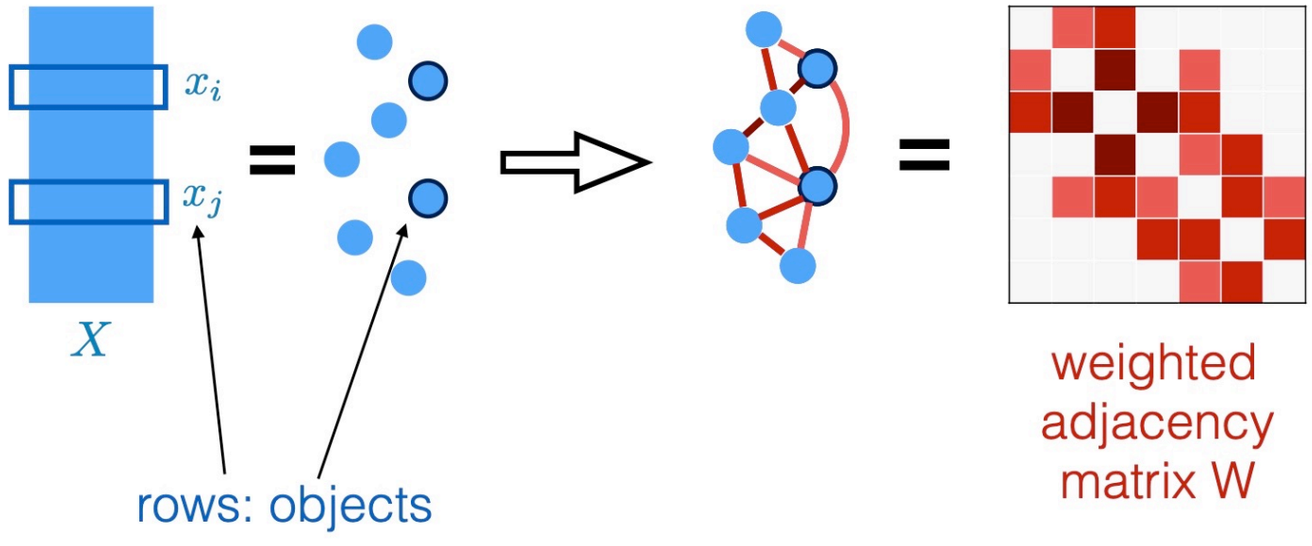
Interlude: you know other methods!

- **2-c) Learn a graph that captures things from the data**

- **The general setting:**
  from observations.... ...find a graph.... ...that models well the data

Given
matrix X

learn
graph G



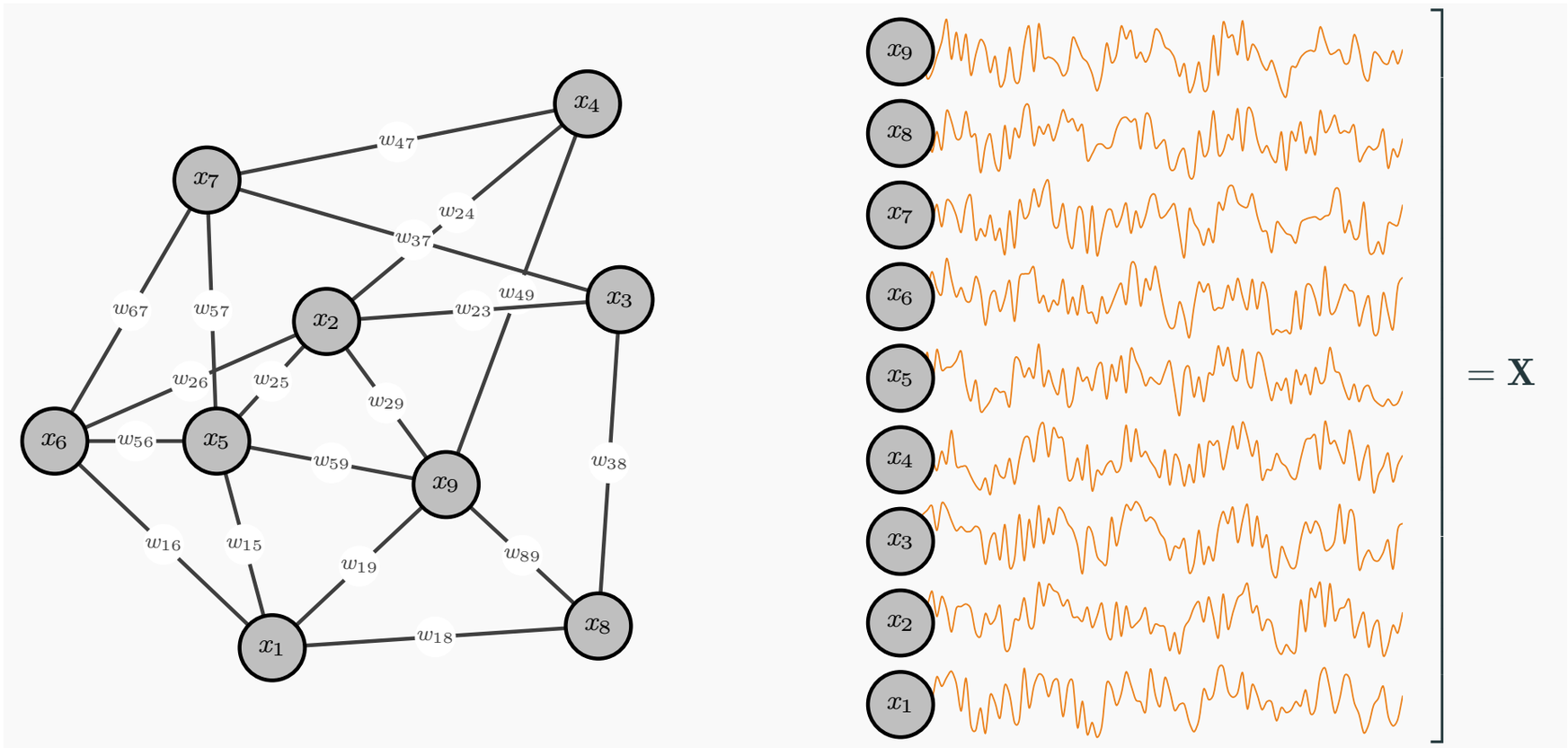
here: some model is useful
e.g.: Gaussian model

- Gaussian Graphical Models
- Bayesian Networks
- Methods with optimization and GSP inside!

Create a graph to represent the data

The general view

(some slides thanks to Bouchard, Breloy, Mian, Hippert-Ferrer)



	Node	Edge
Social network	Person	Relationship
Embedded systems	Sensor	Communication channel
Finance	Company	Ownership
Molecular graph	Atom	Chemical bond
Language	Word	Semantic link
Public transport	Station	Active line
Internet	Pages	Link
Citation	Paper	Citations
Neuroscience	EEG sensor	Brain connectivity

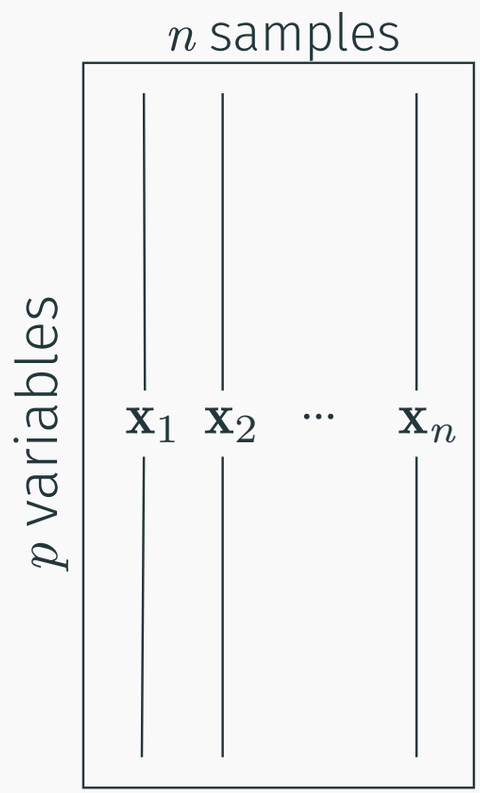
Create a graph to represent the data

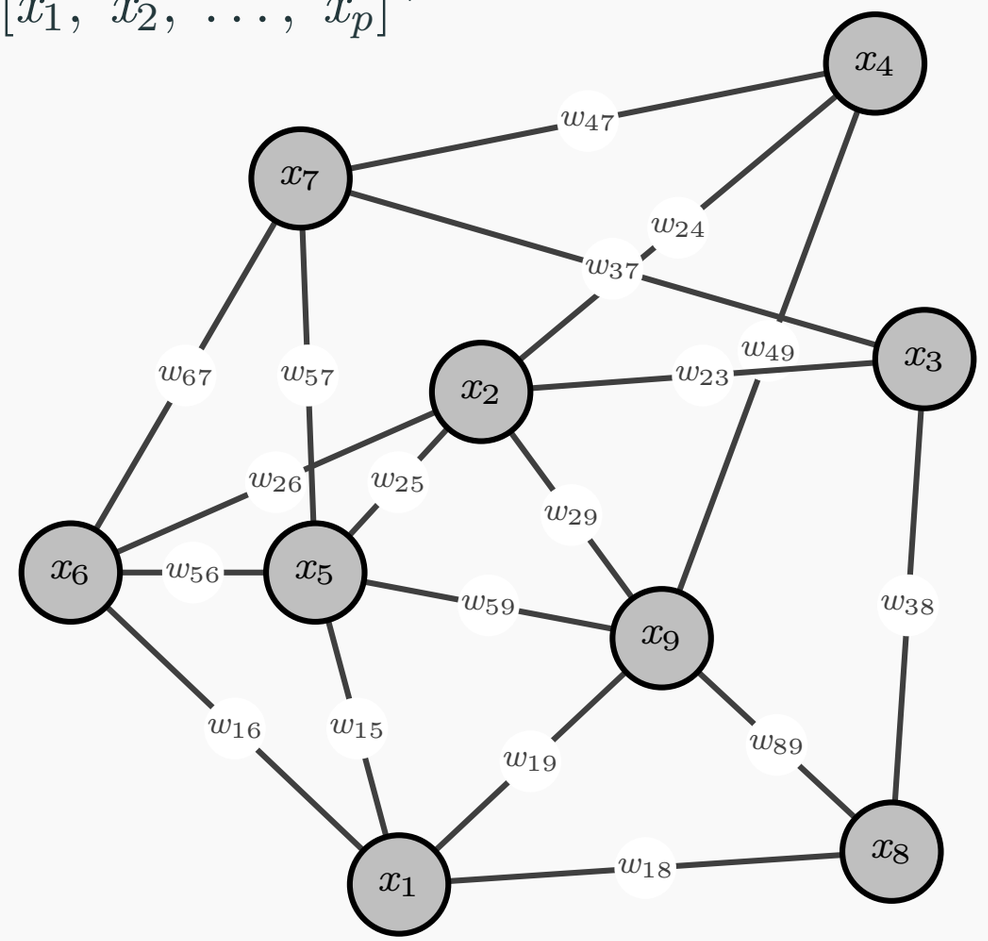
The general view of Graph learning

(some slides thanks to Bouchard, Breloy, Mian, Hippert-Ferrer)

When the **graph topology is unknown** but **each node generates data**

Learn the underlying graph from samples $\mathbf{x} = [x_1,\ x_2,\ \ldots,\ x_p]^\top$





Create a graph to represent the data

Graph learning: with graphical model

An **edge** encodes the "***relationship***" between two **nodes**

We can consider a **statistical definition of relationship** [Dem72; Lau96]

"Connection in the graph = conditional dependence"

The **conditional dependence** $\neg(x_1 \perp\!\!\!\perp x_2)$ holds if

$$P(x_1|x_2, \underbrace{x_3, \cdots, x_p}_{\mathbf{x}_\perp}) \neq P(x_1|\mathbf{x}_\perp)$$

"x_2 adds information to $\mathbf{x}_\perp$ in order to predict x_1"

Independance and conditional independence

Let us take A, B et C three random variables.

Independence

A and B are independent

$\Leftrightarrow$

$P(A, B) = P(A)P(B)$

Notation: $A \perp\!\!\!\perp B$

Conditional independence

A and B are conditionnaly independent to C

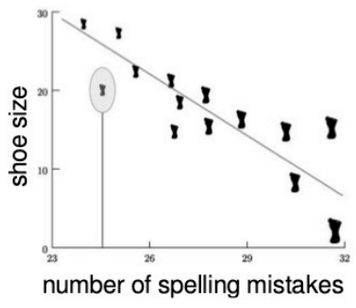
$\Leftrightarrow$

$P(A, B|C) = P(A|C)P(B|C)$

Notation: $A \perp\!\!\!\perp B|C$

Different interpretation - example 1

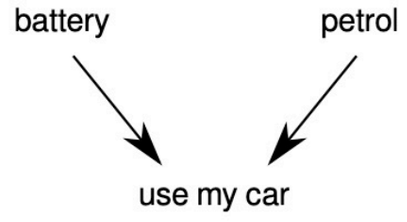
From N. Gauvrit *Statistiques: méfiez-vous*



A= shoe size B = number of spelling mistakes C = age

$A \not\perp\!\!\!\perp B$ but $A \perp\!\!\!\perp B|C$

Different interpretation - example 2



A= battery B = petrol C = use my car

$A \perp\!\!\!\perp B$ but $A \not\perp\!\!\!\perp B|C$

Create a graph to represent the data

Graph learning: with graphical model

Gaussian graphical models

Gaussian graphical model:

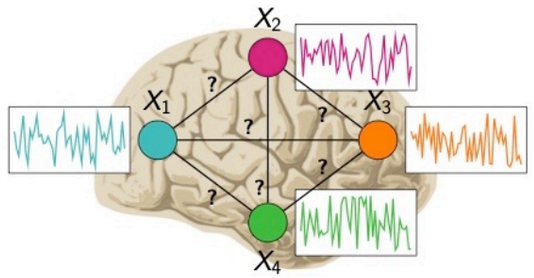
$$(\mathbf{X}, G)$$

$\mathbf{X}$: Gaussian multivariate process
Σ: covariance matrix of $\mathbf{X}$
G: conditional independence structure of $\mathbf{X}$
$K = inv(\Sigma)$: precision matrix

$$i \in V\text{: variable } X_i$$

$$(i,j) \notin E \Leftrightarrow X_i \perp\!\!\!\perp X_j | X_{V \setminus \{i,j\}}$$
$$\Leftrightarrow (K)_{ij} = 0$$

Graphical representation hypothesis



$$\mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix}$$

$G = (V, E)$?

V: set of nodes of graph G — E: set of edges

→ instantaneous interactions — → conditional independence

Create a graph to represent the data

Graph learning: with graphical model

Theorem

Let

$$\mathbb{E}[\mathbf{x}\mathbf{x}^\top] = \boldsymbol{\Sigma} = \boldsymbol{\Theta}^{-1}$$

Covariance *Precision*

Then

$$x_q \perp\!\!\!\perp x_\ell \quad \Leftrightarrow \quad \boldsymbol{\Theta}_{q\ell} = 0$$

no edge $w_{q\ell}$ on the graph *null $q\ell^{th}$-element in the precision*

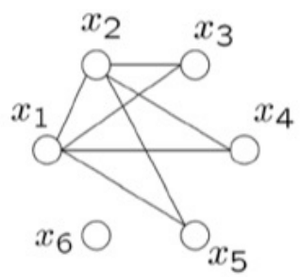
Holds for

- **Gaussian model**
- Most **Elliptical distributions**
- **Semiparametric Gaussian copula** aka “nonparanormal”

Key property of GGM: **the precision matrix is 0 iif the partial correlation coefficient is 0**

$$Q_{i,j} = 0 \quad \Rightarrow \quad X_i \perp X_j | \mathbf{X}_{-ij} \quad \text{or} \quad p(X_i, X_j | \mathbf{X}_{-ij}) = p(X_i | \mathbf{X}_{-ij}) p(X_j | \mathbf{X}_{-ij})$$

$$Q = \begin{pmatrix} * & * & * & * & * & 0 \\ * & * & * & * & * & 0 \\ * & * & * & 0 & 0 & 0 \\ * & * & 0 & * & 0 & 0 \\ * & * & 0 & 0 & * & 0 \\ 0 & 0 & 0 & 0 & 0 & * \end{pmatrix}$$



Create a graph to represent the data

Graph learning: with graphical model

A **Gaussian graphical model** implies a **sparse precision matrix** $\Theta = \Sigma^{-1}$

=> threshold the precision matrix ?

=> covariance selection of Dempster, 1972

- Estimate $\Sigma = \mathbb{E}\left[\boldsymbol{x}\boldsymbol{x}^\top\right]$ from $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\mathbf{0}, \Sigma)$
- Explore sparsity of Σ^{-1}
- $\Sigma^{-1}_{i,j} = 0$ encodes conditional independence relationships

=> multiple test for covariance selection
[Drton, Perlman, 2004]

=> penalised maximum likelihood estimation with a sparsity term for K

=> probabilistic approaches for p(G|X)

Graphical Lasso (GLasso) $\Leftrightarrow$ regularized MLE of Θ

$$\underset{\Theta \in \mathcal{S}_p^{++}}{\text{minimize}} \quad \underbrace{\text{Tr}\{\mathbf{S}\Theta\} - \log\det(\Theta)}_{\text{Gaussian log-likelihood}} + \underbrace{\lambda\|\Theta\|_{1,\text{off}}}_{\text{sparse penalty}}, \qquad \mathbf{S} = \frac{1}{n}\mathbf{X}\mathbf{X}^\top$$

$\longrightarrow$ Graph drawn from Θ's support

Create a graph to represent the data

Graph learning: with graphical model

Solution $\boldsymbol{\Theta}$ such that: $-\boldsymbol{\Theta}^{-1} + \mathbf{S} + \lambda\boldsymbol{\Gamma} = \mathbf{0}$,

where $\boldsymbol{\Gamma}$ is a matrix of element-wise signs of $\boldsymbol{\Theta}$, *i.e.*,

$\Gamma_{ij} = \text{sign}(\Theta_{ij})$ if $\Theta_{ij} \neq 0$, $\Gamma_{ij} \in [-1, 1]$ if $\Theta_{ij} = 0$

Positivity yields: $W_{ii} = S_{ii} + \lambda$, where $\mathbf{W} = \boldsymbol{\Theta}^{-1}$

GLasso based on **block-coordinate method**:

$$\begin{pmatrix} \boldsymbol{\Theta}_{11} & \boldsymbol{\Theta}_{12} \\ \boldsymbol{\Theta}_{21} & \Theta_{22} \end{pmatrix} \qquad \begin{pmatrix} \boldsymbol{\Gamma}_{11} & \boldsymbol{\Gamma}_{12} \\ \boldsymbol{\Gamma}_{21} & \Gamma_{22} \end{pmatrix}$$

$\mathbb{R}^{(p-1)\times(p-1)}$, $\mathbb{R}^{(p-1)\times 1}$, $\mathbb{R}$

$$\begin{pmatrix} \mathbf{W}_{11} & \mathbf{W}_{12} \\ \mathbf{W}_{21} & W_{22} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\Theta}_{11}^{-1} + \frac{\boldsymbol{\Theta}_{11}^{-1}\boldsymbol{\Theta}_{12}\boldsymbol{\Theta}_{21}\boldsymbol{\Theta}_{11}^{-1}}{\Theta_{22} - \boldsymbol{\Theta}_{21}\boldsymbol{\Theta}_{11}^{-1}\boldsymbol{\Theta}_{12}} & -\frac{\boldsymbol{\Theta}_{11}^{-1}\boldsymbol{\Theta}_{12}}{\Theta_{22} - \boldsymbol{\Theta}_{21}\boldsymbol{\Theta}_{11}^{-1}\boldsymbol{\Theta}_{12}} \\ -\frac{\boldsymbol{\Theta}_{21}\boldsymbol{\Theta}_{11}^{-1}}{\Theta_{22} - \boldsymbol{\Theta}_{21}\boldsymbol{\Theta}_{11}^{-1}\boldsymbol{\Theta}_{12}} & \frac{1}{\Theta_{22} - \boldsymbol{\Theta}_{21}\boldsymbol{\Theta}_{11}^{-1}\boldsymbol{\Theta}_{12}} \end{pmatrix}$$

Create a graph to represent the data

Graph learning: with graphical model

For the considered block: $\boldsymbol{\Theta}_{11}^{-1}\boldsymbol{\Theta}_{12}\, W_{22} + \mathbf{S}_{12} + \lambda\boldsymbol{\Gamma}_{12} = \mathbf{0}$

Equivalent to: $\underset{\boldsymbol{\alpha}\in\mathbb{R}^{p-1}}{\text{minimize}} \quad \frac{1}{2}\boldsymbol{\alpha}^\top\boldsymbol{\Theta}_{11}^{-1}\boldsymbol{\alpha} + \boldsymbol{\alpha}^\top\mathbf{S}_{12} + \lambda\|\boldsymbol{\alpha}\|_1$

Then: $\boldsymbol{\Theta}_{12} = \alpha/\, W_{22}$ $\Theta_{22} = \frac{1}{W_{22}} + \boldsymbol{\Theta}_{21}\boldsymbol{\Theta}_{11}^{-1}\boldsymbol{\Theta}_{12}$

From there:

- $\boldsymbol{\Theta}_{11}^{-1} = \mathbf{W}_{11} - \mathbf{W}_{12}\mathbf{W}_{21}/\, W_{22}$
- Update $\mathbf{W}$ through the block identity

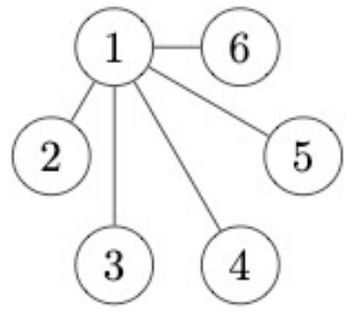
Algorithm

1. Initialize $\mathbf{W} = \text{diag}(\mathbf{S}) + \lambda\mathbf{I}_p$
2. While not convergence, cycle around the columns:
   a. Rearrange rows/columns so that target one is last
   b. Compute $\boldsymbol{\Theta}_{11}^{-1} = \mathbf{W}_{11} - \mathbf{W}_{12}\mathbf{W}_{21}/\, W_{22}$
   c. Solve $\underset{\boldsymbol{\alpha}\in\mathbb{R}^{p-1}}{\text{minimize}} \quad \frac{1}{2}\boldsymbol{\alpha}^\top\boldsymbol{\Theta}_{11}^{-1}\boldsymbol{\alpha} + \boldsymbol{\alpha}^\top\mathbf{S}_{12} + \lambda\|\boldsymbol{\alpha}\|_1$
   d. Update $\boldsymbol{\Theta}_{12} = \alpha/\, W_{22}$ and $\Theta_{22} = \frac{1}{W_{22}} + \boldsymbol{\Theta}_{21}\boldsymbol{\Theta}_{11}^{-1}\boldsymbol{\Theta}_{12}$
   e. Update $\boldsymbol{\Theta}$ and $\mathbf{W}$ from block identity, ensuring $\boldsymbol{\Theta}\mathbf{W} = \mathbf{I}_p$
3. Output **precision** $\boldsymbol{\Theta}$ and **covariance** $\mathbf{W}$

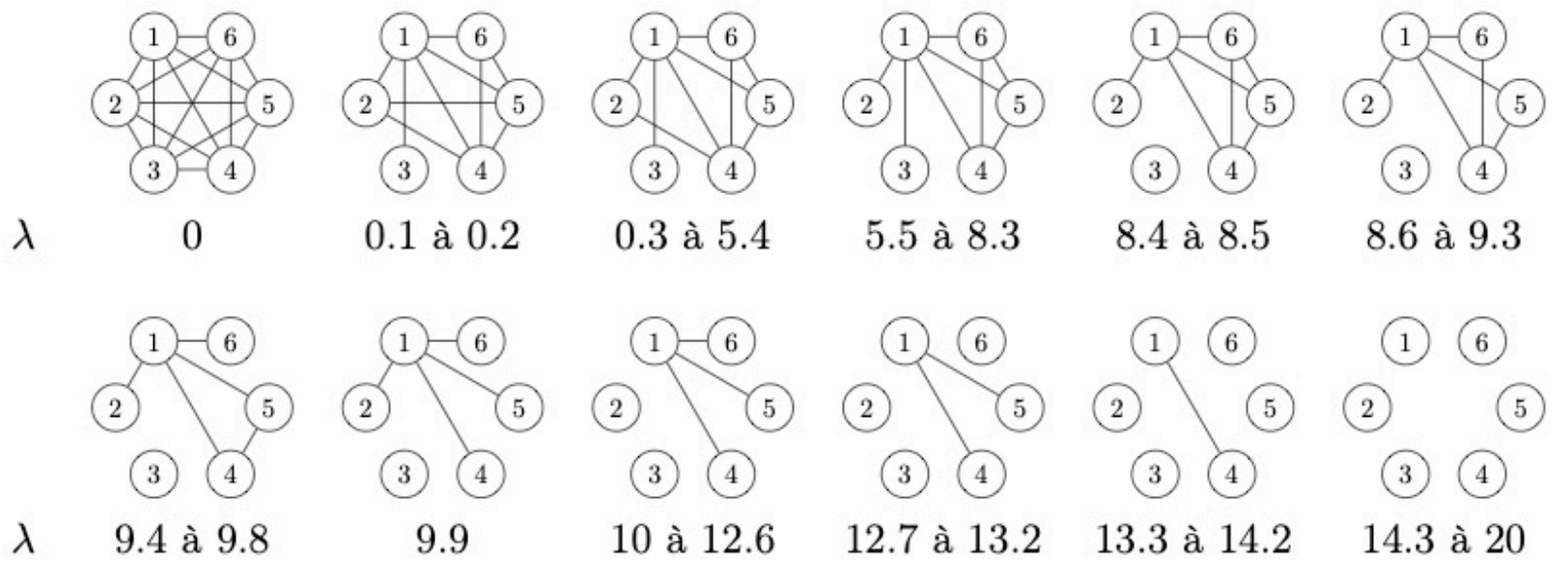
Create a graph to represent the data

Graph learning: with graphical model

EXAMPLE [Costard, 2014]: we generate 600 observations according to the GMM



Then, we use graphical lasso, with varying penalisation parameter



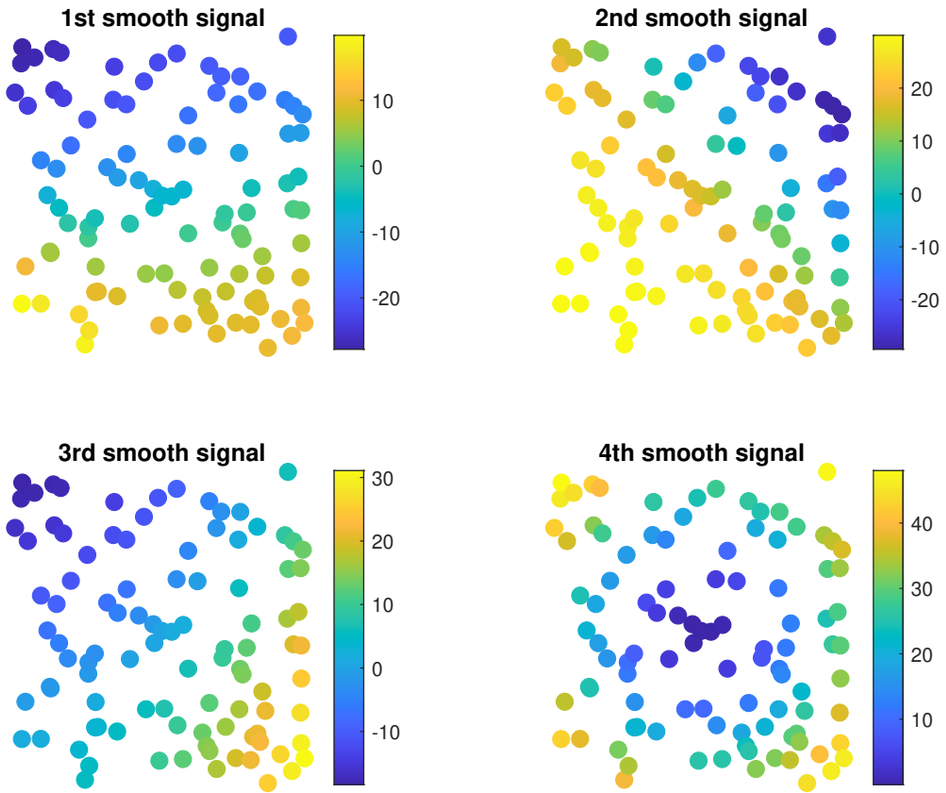
Next question is: How to choose λ ? (See Costard PhD thesis, and ask Titouan V. for more recent answer)

Create a graph to represent the data

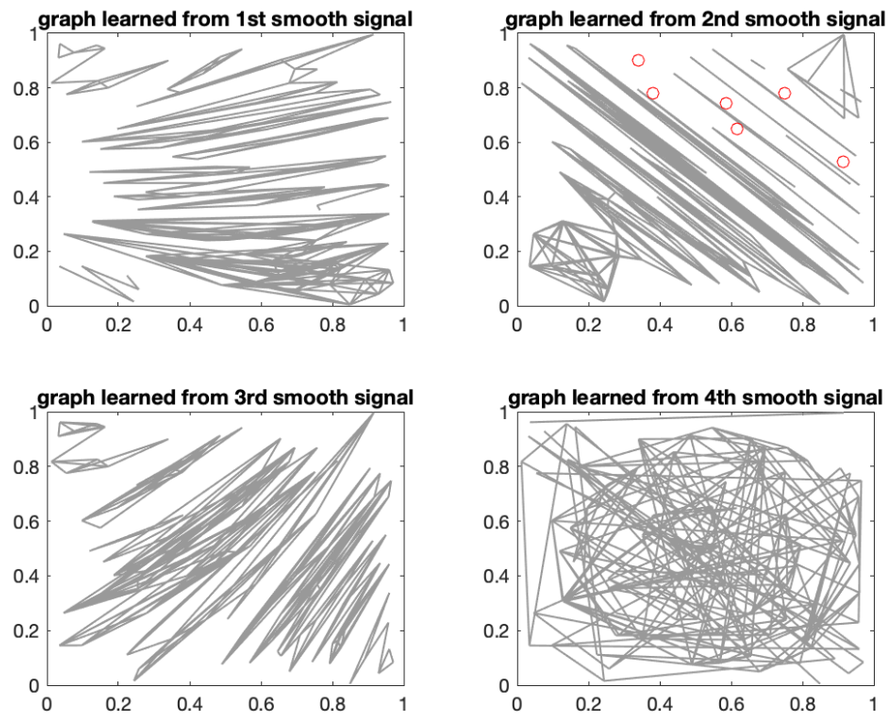
Graph learning: with GSP

- Connecting the dots: infer networks from GSP as in Segarra et al. (2017); Pasdeloup et al. (2018); Mateos et al. (2019)
- Learning of Laplacian or Adjacency matrices from a constraint of **smoothness of the data**, operator constraints, structural constraints, spectral constraints, as in (Kalofolias, 2016; Dong et al., 2016; Thanou et al., 2017; Egilmez et al., 2017; Dong et al., 2019),... (+ Apologies to all the missing references)

In images, from (Kalofolias, 2016):



into 4 graphs:



Create a graph to represent the data

Graph learning: with GSP

Smoothness of a graph signal measured by **graph signal variation**

$$\mathrm{GSV}(\mathbf{x}) = \sum_{q,\ell} A_{q,\ell}(x_q - x_\ell)^2 = \mathbf{x}^\top \mathbf{L}\mathbf{x}$$

Total GSV for a **data matrix** $\mathbf{X}$

$$\sum_{i=1}^{n} \mathbf{x}_i^\top \mathbf{L}\mathbf{x}_i = \mathrm{Tr}(\mathbf{X}^\top \mathbf{L}\mathbf{X}) \propto \mathrm{Tr}(\mathbf{L}\mathbf{S}) \qquad \text{with } \mathbf{S} = \mathbf{X}\mathbf{X}^\top/n$$

Could we **learn the graph** that yields the "*smoothest*" observation $\mathbf{X}$?

$$\underset{\mathbf{L}\in\mathcal{L}}{\text{minimize}} \quad \mathrm{Tr}(\mathbf{X}^\top \mathbf{L}\mathbf{X}) \qquad \Rightarrow \qquad \text{trivial solution } \mathbf{L} = \mathbf{0}$$

$\rightarrow$ **Solution 1**: design of fitting-penalties to get meaningful solutions

$\rightarrow$ **Solution 2**: Link the problem to gaussian graphical models

Create a graph to represent the data

Graph learning: with GSP

A generic problem formulation

$$\underset{\mathbf{L}\in\mathcal{L}}{\text{minimize}} \quad \underbrace{\text{Tr}(\mathbf{X}^\top\mathbf{L}\mathbf{X})}_{\text{Smothness}} + \overbrace{f_+(\mathbf{L})}^{\text{avoid } \mathbf{L}=\mathbf{0}} + \underbrace{f_{\text{sp}}(\mathbf{L})}_{\text{promotes sparsity}}$$

- Some works equivalently formalized the problem *wrt* $\mathbf{A}$
- Some works considered constraints instead of penalty f_+
- The construction and motivation for each terms evolved between 2016-2023

From Kalofolias 2016:

Promoting smoothness = graph sparsity

$$\mathbf{Z} \in \mathbb{R}^{p\times p} : \|\mathbf{x}_i - \mathbf{x}_j\|_2^2, \qquad \text{Tr}(\mathbf{X}^\top\mathbf{L}\mathbf{X}) = \frac{1}{2}\text{Tr}(\mathbf{A}\mathbf{Z}) = \frac{1}{2}\underbrace{\|\mathbf{A}\odot\mathbf{Z}\|_1}_{\text{weighted } \ell\text{-1 norm}}$$

$$\text{Tr}(\mathbf{X}^\top\mathbf{L}\mathbf{X}) + \lambda\|\mathbf{A}\|_1 = \frac{1}{2}\|\mathbf{A}\odot(2\lambda\mathbf{1}\mathbf{1}^\top + \mathbf{Z})\|_1$$

$\rightarrow$ Adding another sparsity term not necessarily useful

Create a graph to represent the data

Graph learning: with GSP

$$\underset{\mathbf{A}\in\mathcal{A}}{\text{minimize}} \quad f(\mathbf{A}) + \|\mathbf{A}\odot\mathbf{Z}\|_1$$

From Kalofolias 2016:

Example 1 – Gaussian kernel graph

$$f(\mathbf{A}) = \sigma^2 \sum_{ij} A_{ij}(\log(A_{ij}) - 1)$$

Solution:

$$A_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{\sigma^2}\right)$$

promotes graph density by penalizing big weights while allowing small ones

Example 2 $f(\mathbf{A}) = \alpha\|\mathbf{A1}\|_2^2 + \alpha\|\mathbf{A}\|_2^2 \quad (= \alpha\|\mathbf{L}\|_2^2\,), \qquad$ subject to $\|\mathbf{A}\|_1 = s$ [Don+16]

force degrees to be positive, do not prevent edges to be zero

Example 3 $f(\mathbf{A}) = \frac{\beta}{2}\|\mathbf{A}\|_2^2 \; -\alpha\mathbf{1}^\top\log(\mathbf{A1}), \quad \alpha > 0, \beta \geq 0$ [Kal16]

promotes graph density

Solve these **optimization problems** → primal dual techniques from [KP15]

Create a graph to represent the data

Graph learning: with GSP

From Kalofolias 2016:

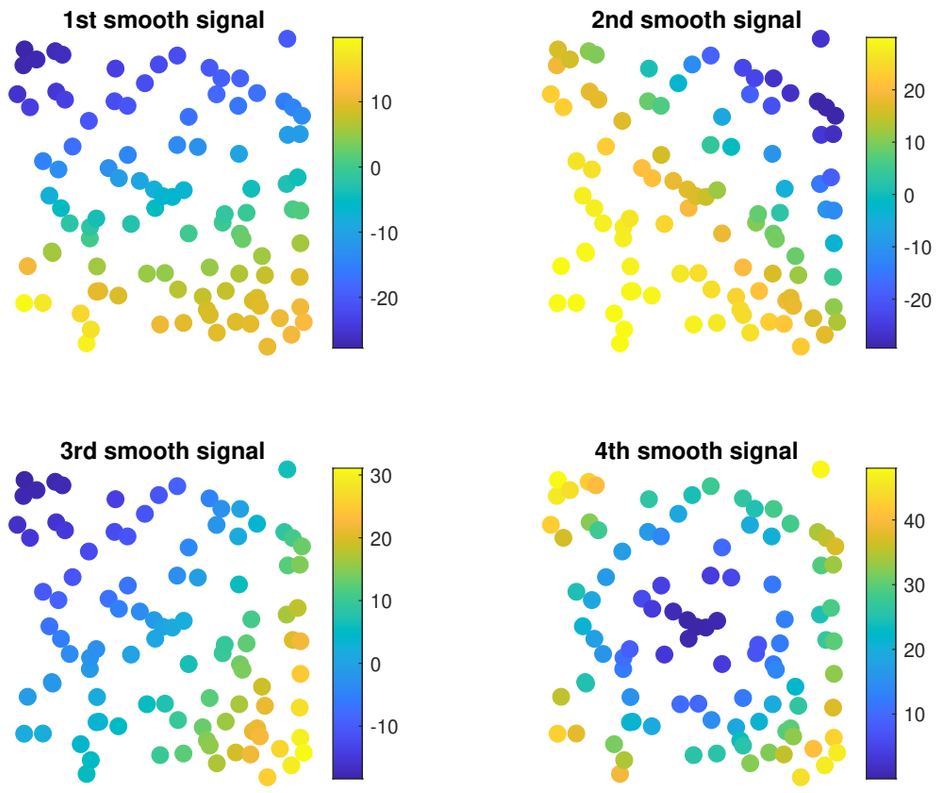
$$\text{(P-stat)} \qquad \min_{W \in \mathscr{W}} f(W) = \min_{W \in \mathscr{W}} \|W \circ Z\|_{\mathbf{1}} - \alpha 1^\top \log(W1) + \beta \|W\|_F^{\mathbf{2}}.$$

Key quantity:
Given $Z_{ij} = \|x_i - x_j\|^{\mathbf{2}}$, the total **global variations** on the graph of $X \in \mathbb{R}^{N \times m}$ (having m components per node, e.g. 12 for a monthly signal spanning a year) are:

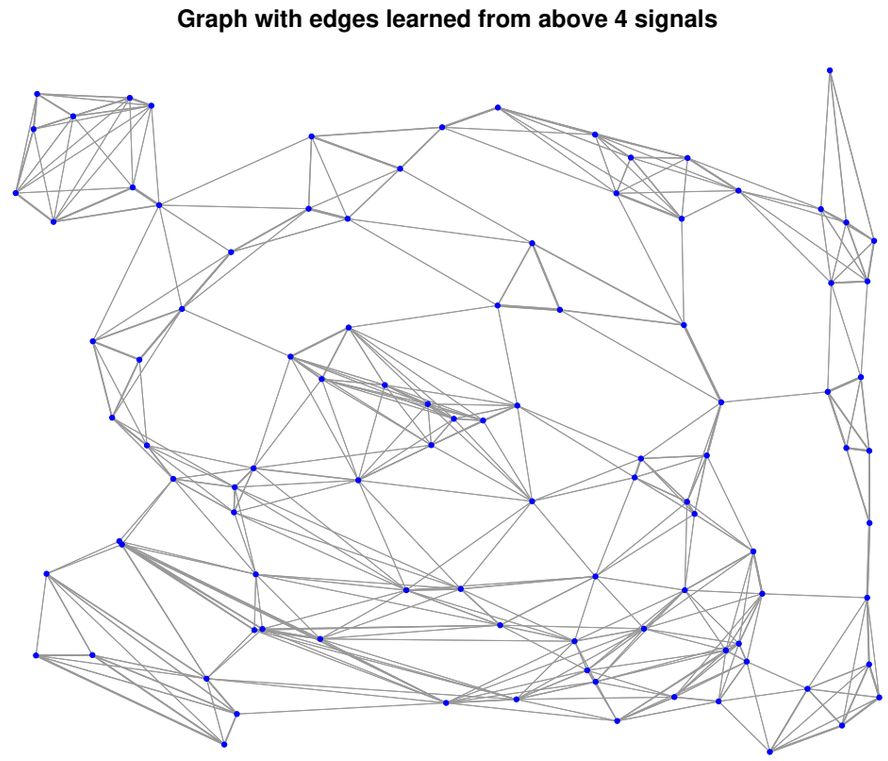
$$\frac{1}{2} \sum_{(i,j) \in V \times V} W_{ij} \|x_i - x_j\|^{\mathbf{2}} = \|W \circ Z\|_{\mathbf{1}}.$$

$\mathscr{W} = \{W \in \mathbb{R}_+^{N \times N} \text{ s.t. } W = W^\top, \mathrm{diag(W)} = 0\}$: admissible set.

- 2nd terms = regularizations =
- control of average degree node, parameter α
- control the density of the graph, parameter β



into 1 graph:



Create a graph to represent the data

Graph learning: with GSP — development 1: Gaussian Markov Random Fields

Combine loss from graphical lasso and behaviour of the Laplacian of a graph:

From Elgimez, Pavel, Ortega, 2017

$$\underset{\mathbf{L}\in\mathcal{L}}{\text{minimize}} \quad \mathrm{Tr}(\mathbf{L}\,\mathbf{S}) - \log\det(\mathbf{L}) + \|\mathbf{L}\odot\mathbf{H}\|_1$$

sample covariance matrix

symmetric regularization matrix

ℓ-1 norm penalty: Since $\forall i \neq j,\ L_{ij} \leq 0,\ L_{ii} > 0$, one can choose $\mathbf{H}$ such that

$$\|\mathbf{L}\odot\mathbf{H}\|_1 = \mathrm{Tr}(\mathbf{L}\mathbf{H})$$

Simple to optimize

$$\underset{\mathbf{L}\in\mathcal{L}}{\text{minimize}} \quad \mathrm{Tr}(\mathbf{L}\mathbf{K}) - \log\det(\mathbf{L}), \qquad \mathbf{K} = \mathbf{S} + \mathbf{H}$$

Create a graph to represent the data

Graph learning: with GSP — development 2: Robust or Structured GL

From Hippert-Ferrer, ECML 2023

Fit graph to data — Promote graph sparsity

$$\underset{\Theta}{\text{minimize}} \quad f(\Theta) + \lambda h(\Theta)$$

$$\text{subject to} \quad \Theta \in \mathcal{S}_\Theta \quad \text{and} \quad \Theta^\dagger = \Sigma \in \mathcal{S}_\Sigma$$

Structure in precision — Structure in covariance

1. Design model f and penalty h

- Elliptical distributions [Cardoso22]
- Gaussian copulas [Liu09]
- Robust fitting [Phi24]
- Nonconvex penalties [Cardoso20]

2. Leverage structures on Θ

- Laplacian constrained [Eğilmez17]
- Spectral structure [Kumar22]
- Factor models [Hippert-ferrer23]
- Total positivity [Lauritzen19]
- $\Theta^\dagger =$ Correlation [Wed, 28 at eusipco 24!]

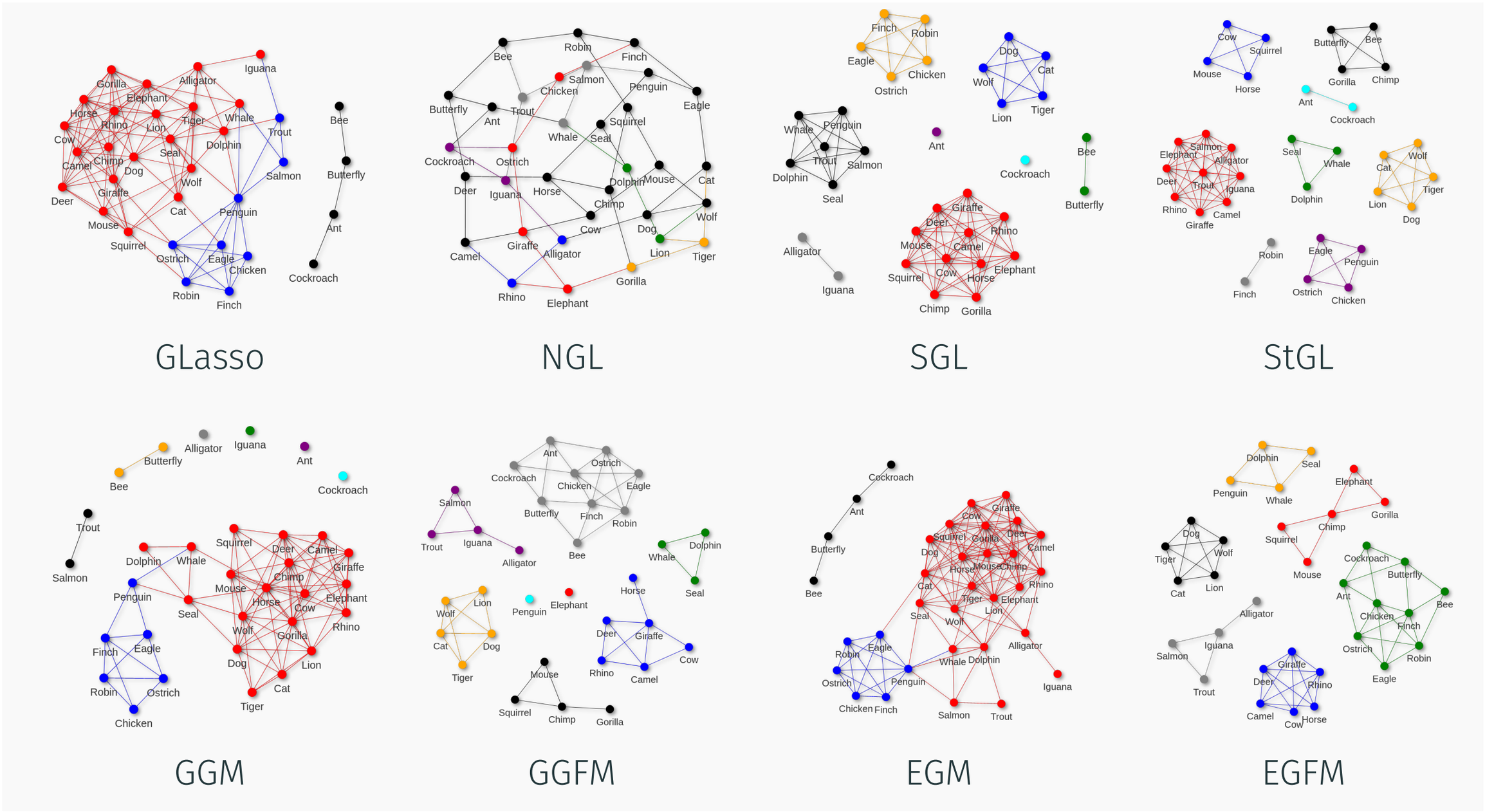
Code

`https://github.com/ahippert/graphfactormodel`

Create a graph to represent the data

Graph learning: with GSP — development 2: Robust or Structured GL

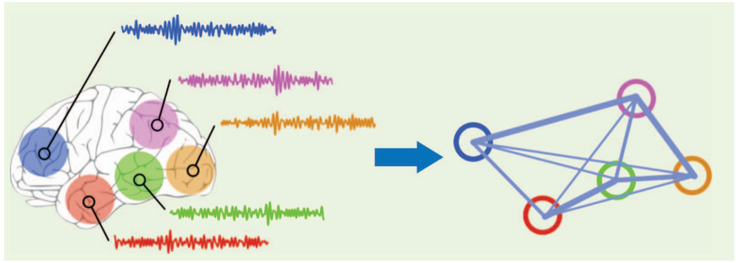
From Hippert-Ferrer, ECML 2023



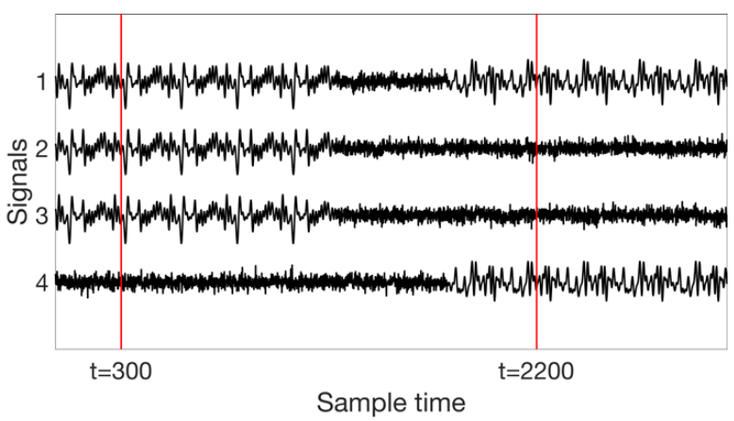
Create a graph to represent the data

Graph learning: with GSP — development 3: time-series

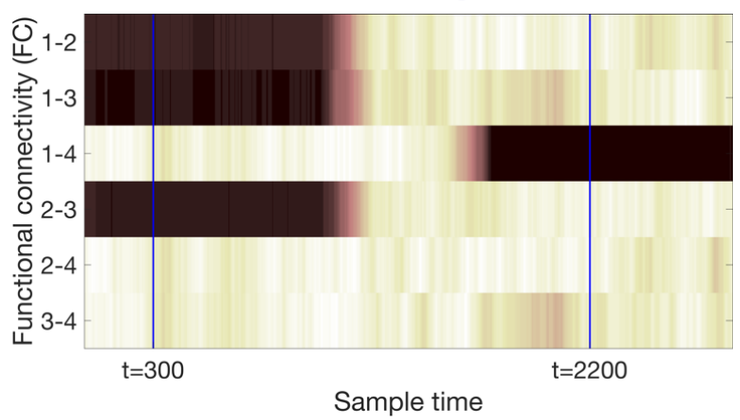
Richiardi et al. (2013)



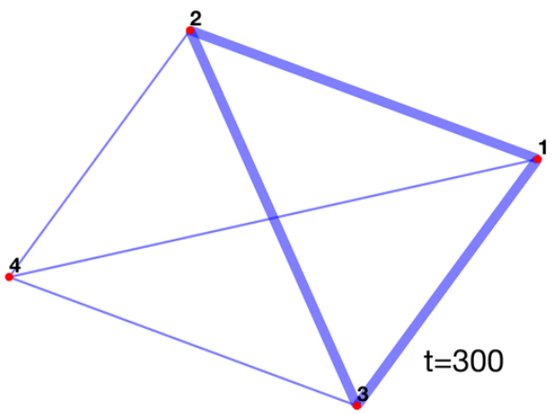
[G. Frusque, 2020]



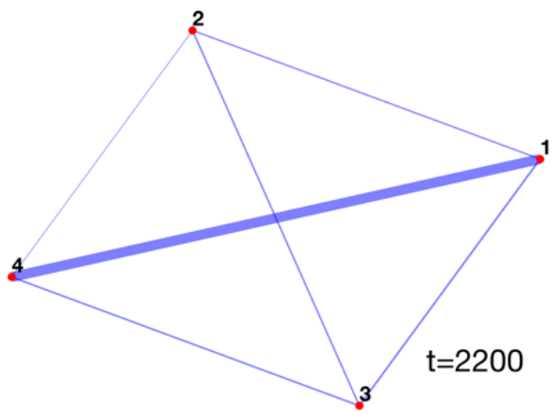
Model of 4 signals with temporal synchrony



Corresponding functional connectivity in function of time samples



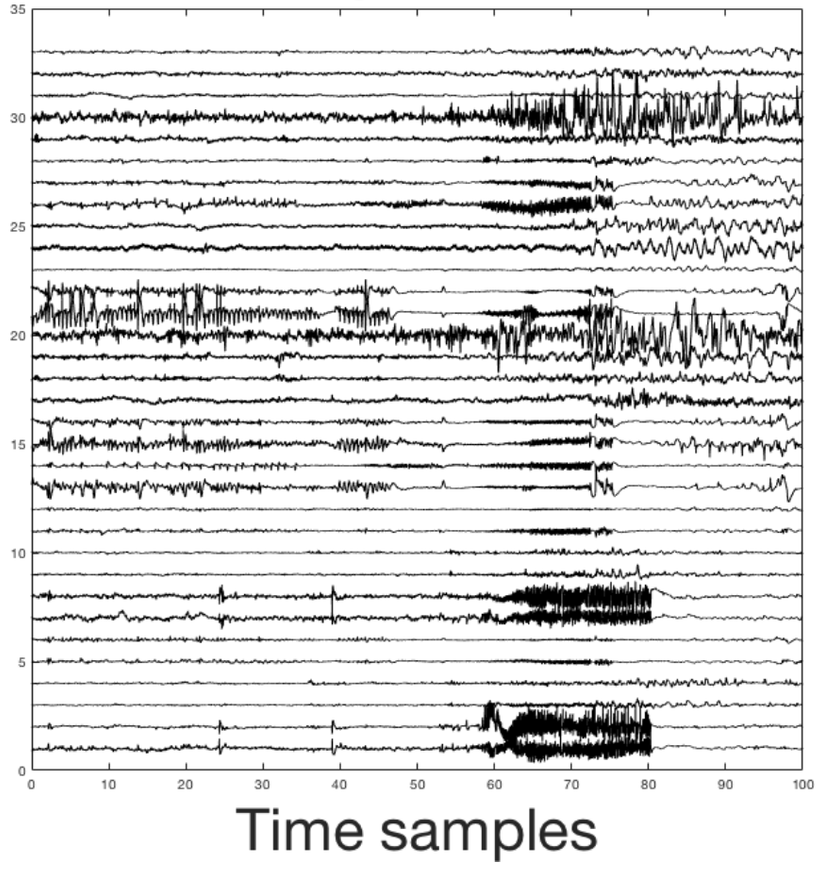
Dynamical graph at $t = 300$



Dynamical graph at $t = 2200$

Recordings of iEEG for Epileptic treatment

Example of recording: a multivariate signal...



Create a graph to represent the data

Graph learning: with GSP — development 3: time-series

The general idea: add a “smoothness” term in time, or “sparsity”

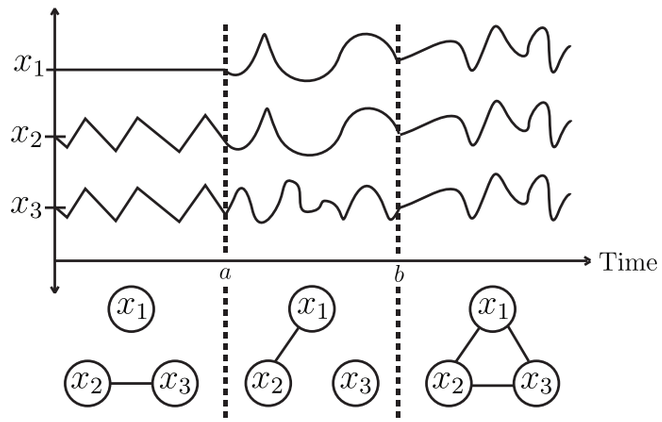


Figure 1: Three sensors with associated time series readings. Based on such data, we infer a time-varying network that reveals 1) the dependencies between the different sensors, and 2) when and how these dependencies change over time.

From Hallac et al., 2017

- Kalofolias et al. (2017): a time-varying setting with smooth variations of the inferred graphs, thanks to a Tikhonov regularization term

$$\sum_{k=\mathbf{2}}^{T} \| W_k - W_{k-\mathbf{1}} \|_F^{\mathbf{2}}$$

- Yamada et al. (2019, 2020) temporal sparsity prior with $\ell_{\mathbf{1}}$ norm better suited to sharp changes

  (see Hallac et al. (2017); Jung et al. (2015) for graph-lasso)
- Yamada et al. (2020): group lasso term (global changes at sparse time points) vs. fused lasso term (local change at sparse time points)

Create a graph to represent the data

Graph learning: with GSP — development 3: time-series

Optimization problem for Dynamic graph learning:
(P-dyn)

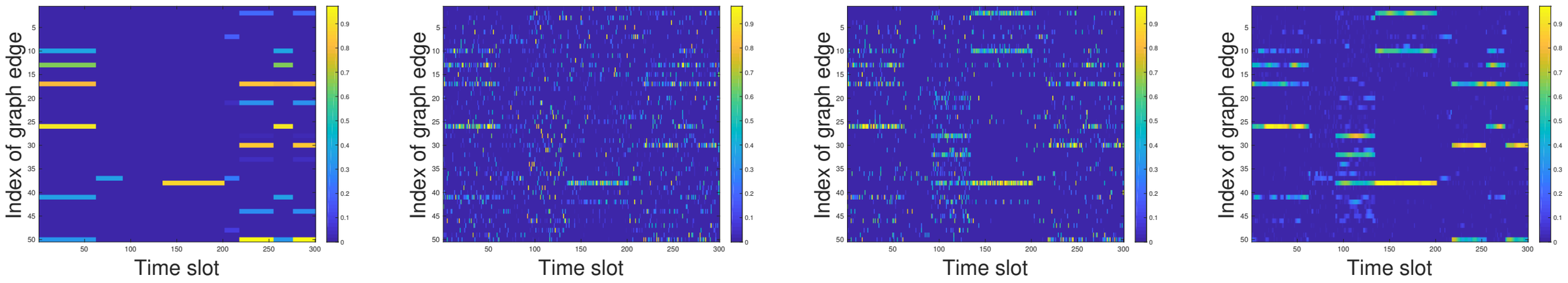
$$\min_{W_k \in \mathscr{W}} \sum_{k=\mathbf{1}}^{T} f_k(W_k) + \eta \sum_{k=\mathbf{2}}^{T} \|W_k - W_{k-\mathbf{1}}\|_{\mathbf{1}},$$

with $f_k(W_k)$ as in (P-stat) with parameters α_k and β_k

Solution of the optimization problem:

- Primal dual optimization algorithm as in Yamada et al. (2019, 2020) using the primal-dual splitting framework of Condat (2013).

From Yamada et al., 2020



(a) Ground truth (b) SGL-GLasso (c) SGL-Smooth (d) TGL-GLTV

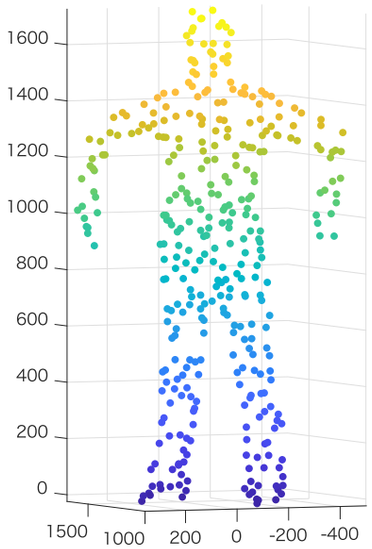
Fig. 5. The visualization of the temporal variations in the time-varying graph learned from the dataset based on the graph that produces large fluctuations at few time slots.

Create a graph to represent the data

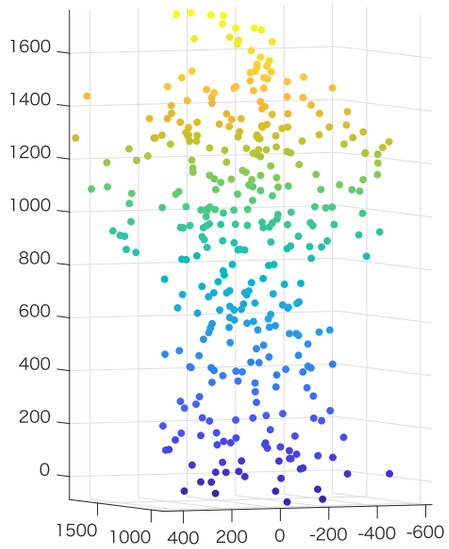
Graph learning: with GSP — development 3: time-series

An example on dynamic point cloud

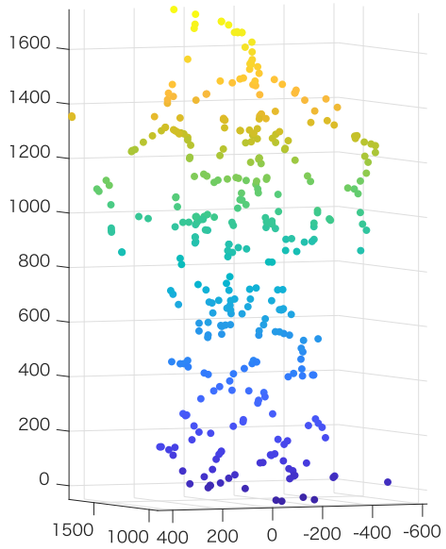
From Yamada et al., 2020



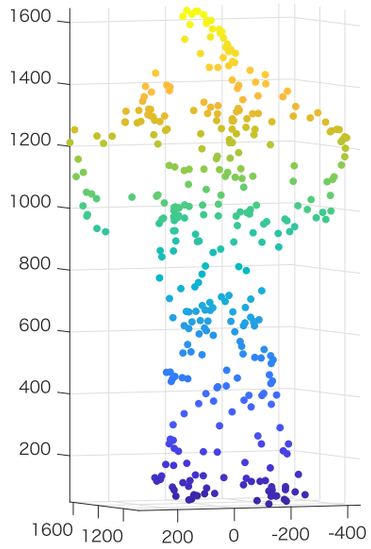
(a) Ground truth



(b) Noisy



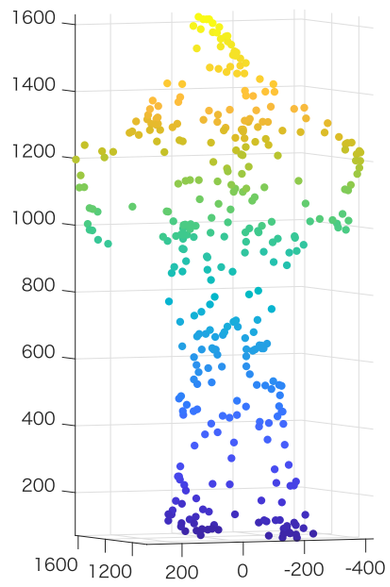
(c) k-nearest neighbor



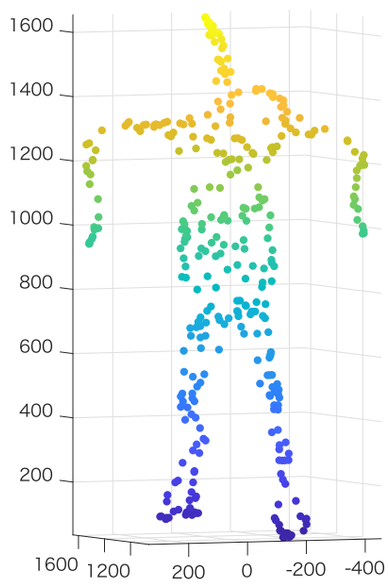
(d) SGL-Smooth

TABLE IV
DENOISING RESULTS

	dog	*handstand*	*skirt*	*wheel*
noisy	12.12	13.97	13.73	15.45
k-NN	12.35	14.29	14.01	15.76
SGL-Smooth	13.13	15.37	15.03	16.74
TGL-Tikhonov	13.34	15.69	15.34	17.06
TGL-FLasso	**20.05**	**21.32**	**21.58**	**22.97**



(e) TGL-Tikhonov



(f) TGL-FLasso

End of the story (?)

Some topics we did not cover (even in G SP+ML):

- Sampling on graphs (of nodes ? of edges ? sampling theorem)
- Stochastic processes on graph and spectral estimation
- Design of filters, wavelets, filter banks,… vertex-frequency,…
- Graph simplification: coarsening, pooling, sparsification
- Applications to images, point clouds,…
- Higher-orders: Hypergraphs, Simplicial Complexes, Hodge Laplacian
- and topics I don’t even know about…

Advertisement: we hire M2 and PhD!

ENS DE LYON

cnrs

Lab PHYS ENS de LYON