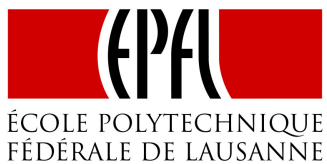
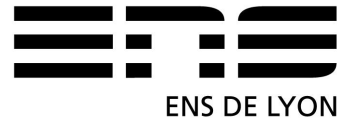
Accelerated Spectral Clustering Using Graph Filtering of Random Signals

Nicolas Tremblay[1,2,3], Gilles Puy[1], Pierre Borgnat[2], Rémi Gribonval[1], Pierre Vandergheynst[1,3]

(1) PANAMA Team, INRIA Rennes, France
(2) Physics Laboratory, Ens de Lyon, CNRS, France
(3) Signal Processing Laboratory 2, EPFL, Switzerland



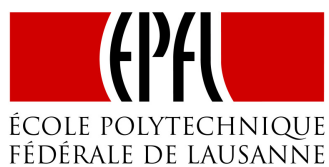


Accelerated Spectral Clustering Using Graph Filtering of Random Signals

Nicolas Tremblay[1,2,3], Gilles Puy[1], Pierre Borgnat[2], Rémi Gribonval[1], Pierre Vandergheynst[1,3]

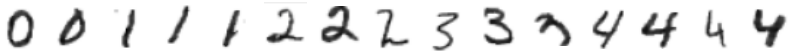
(1) PANAMA Team, INRIA Rennes, France
(2) Physics Laboratory, Ens de Lyon, CNRS, France
(3) Signal Processing Laboratory 2, EPFL, Switzerland

"*Compressive spectral embedding : sidestepping the SVD*", NIPS '15
D. Ramasamy and **U. Madhow**, UC Santa Barbara

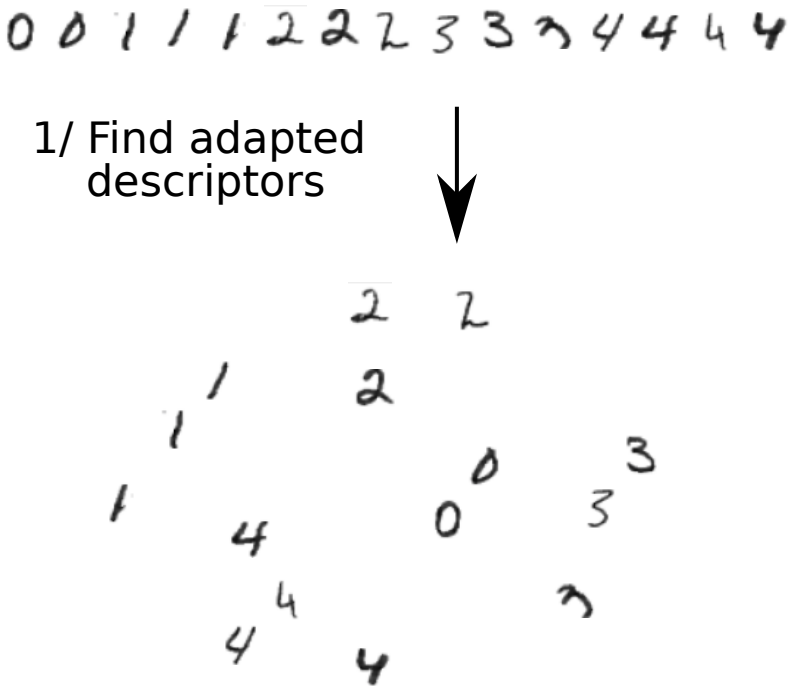


What's clustering ?

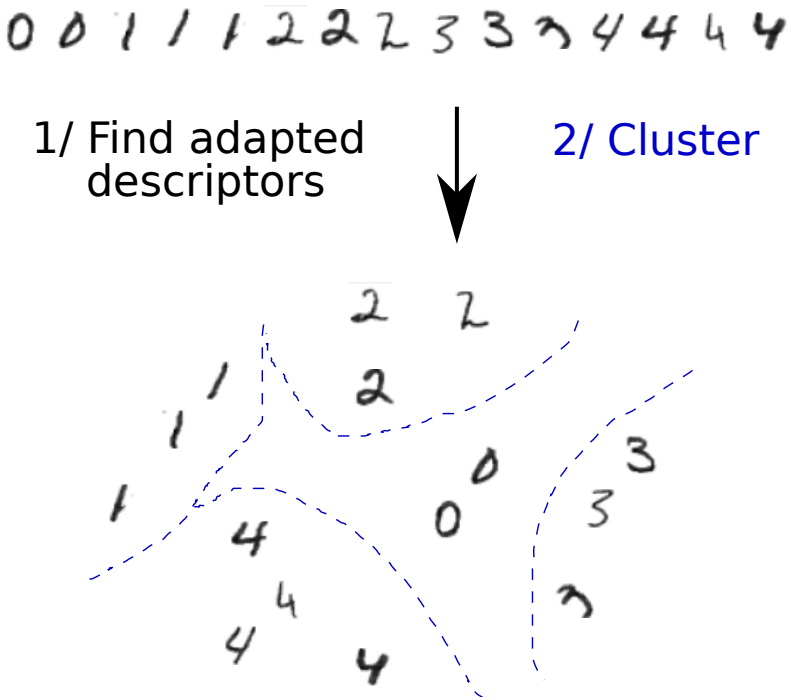
Given a series of N objects :



Given a series of N objects :



Given a series of N objects :



After step 1, one has :

- N vectors in d dimensions (descriptor dimension) :

$$\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N \in \mathbb{R}^d$$

- and their *distance matrix* $\boldsymbol{\Delta} \in \mathbb{R}^{N \times N}$.

After step 1, one has :

- N vectors in d dimensions (descriptor dimension) :

$$\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N \in \mathbb{R}^d$$

- and their *distance matrix* $\boldsymbol{\Delta} \in \mathbb{R}^{N \times N}$.

Goal of clustering : assign a label $c(i) = 1, \cdots, k$ to each object i in order to organize / simplify / analyze the data.

After step 1, one has :

- N vectors in d dimensions (descriptor dimension) :

$$\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N \in \mathbb{R}^d$$

- and their *distance matrix* $\boldsymbol{\Delta} \in \mathbb{R}^{N \times N}$.

Goal of clustering : assign a label $c(i) = 1, \cdots, k$ to each object i in order to organize / simplify / analyze the data.

There exists two different general types of methods :

- methods directly based on the $\boldsymbol{x}_i$ and/or $\boldsymbol{\Delta}$ like k-means or hierarchical clustering.
- graph-based methods.

Graph construction from the distance matrix Δ

Create a graph $\mathcal{G} = (V, E)$:

- each node in V is one of the N objects
- each pair of nodes (i, j) is connected if the associated distance $\Delta(i, j)$ is small enough.

Graph construction from the distance matrix Δ

Create a graph $\mathcal{G} = (V, E)$:

- each node in V is one of the N objects
- each pair of nodes (i, j) is connected if the associated distance $\Delta(i, j)$ is small enough.

For example, two connectivity possibilities :

- Gaussian kernel :
  1. all pairs of nodes are connected with links of weights $\exp(-\Delta(i, j)/\sigma)$
  2. remove all links of weight inferior to ϵ
- k nearest neighbors : connect each node to its k nearest neighbors.

The problem now states :

Given the graph $\mathcal{G}$ representing the similarity between the N objects, find a partition of all nodes in k clusters.

The problem now states :

Given the graph $\mathcal{G}$ representing the similarity between the N objects, find a partition of all nodes in k clusters.

Many methods exist [Fortunato '10] :

- Modularity (or other cost-function) optimisation methods [Newman '06]
- Random walk methods [Schaub '12]
- Methods inspired from statistical physics [Krzakala '13], information theory [Rosvall '08]...
- spectral methods
- ...

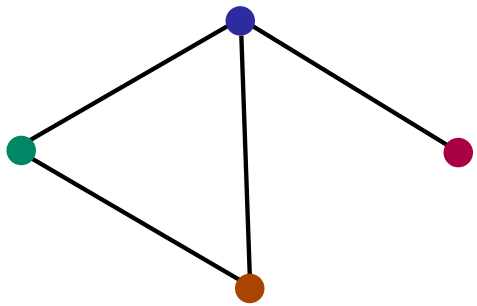
Three useful matrices

The adjacency matrix :

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

The degree matrix :

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



The Laplacian matrix :

$$L = S - W = \begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$

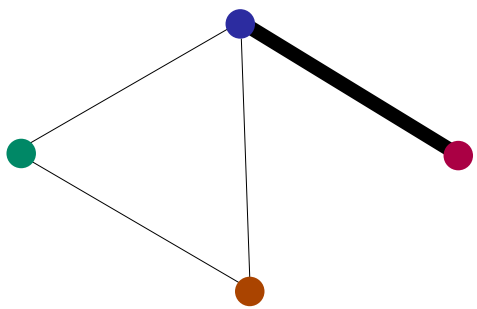
Three useful matrices

The adjacency matrix :

$$W = \begin{bmatrix} 0 & \mathbf{.5} & \mathbf{.5} & 0 \\ \mathbf{.5} & 0 & \mathbf{.5} & \mathbf{4} \\ \mathbf{.5} & \mathbf{.5} & 0 & 0 \\ 0 & \mathbf{4} & 0 & 0 \end{bmatrix}$$

The degree matrix :

$$S = \begin{bmatrix} \mathbf{1} & 0 & 0 & 0 \\ 0 & \mathbf{5} & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & \mathbf{4} \end{bmatrix}$$



The Laplacian matrix :

$$L = S - W = \begin{bmatrix} \mathbf{1} & \mathbf{-.5} & \mathbf{-.5} & 0 \\ \mathbf{-.5} & \mathbf{5} & \mathbf{-.5} & \mathbf{-4} \\ \mathbf{-.5} & \mathbf{-.5} & \mathbf{1} & 0 \\ 0 & \mathbf{-4} & 0 & \mathbf{4} \end{bmatrix}$$

The classical spectral clustering algorithm [Von Luxburg '06] :

Given the N-node graph $\mathcal{G}$ of laplacian matrix L :

1. Compute L's first k eigenvectors :

$$\mathrm{U}_k = (\boldsymbol{u}_1 | \boldsymbol{u}_2 | \cdots | \boldsymbol{u}_k) .$$

2. Consider each node i as a point in $\mathbb{R}^k$:

$$\boldsymbol{f}_i = \mathrm{U}_k^\top \boldsymbol{\delta}_i .$$

3. Run k-means with the Euclidean distance : $D_{ij} = \|\boldsymbol{f}_i - \boldsymbol{f}_j\|$ and obtain k clusters.

The classical spectral clustering algorithm [Von Luxburg '06] :

Given the N-node graph $\mathcal{G}$ of laplacian matrix L :

1. Compute L's first k eigenvectors :

$$\mathrm{U}_k = (\boldsymbol{u}_1 | \boldsymbol{u}_2 | \cdots | \boldsymbol{u}_k) .$$

2. Consider each node i as a point in $\mathbb{R}^k$:

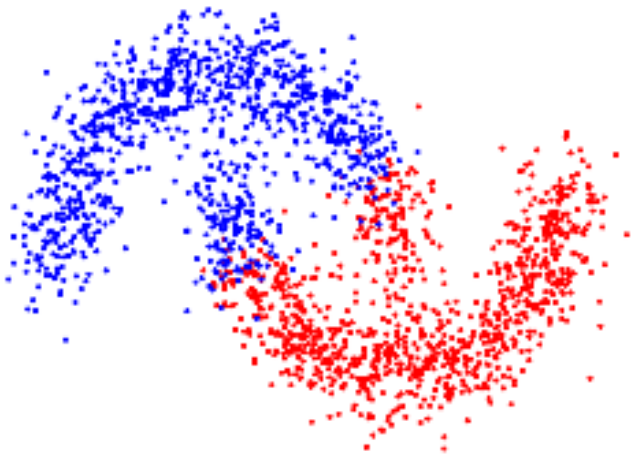
$$\boldsymbol{f}_i = \mathrm{U}_k^\top \boldsymbol{\delta}_i .$$

3. Run k-means with the Euclidean distance : $D_{ij} = \|\boldsymbol{f}_i - \boldsymbol{f}_j\|$ and obtain k clusters.

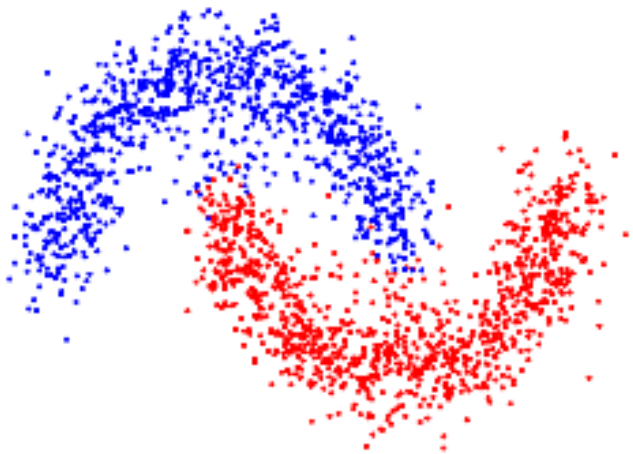
Definition : Let us call D_{ij} the *spectral clustering distance*.

What's the point of using a graph ?

N points in $d = 2$ dims. Result with k-means (k=2) on $\mathbf{\Delta}$:



After creating a graph, partial daigonalisation of L and running k-means (k=2) on $\boldsymbol{D}$:



Overview

Problem : When N and/or k become too large, there are two main bottlenecks in the algorithm :

1. The partial eigendecomposition of the (sparse) Laplacian with restarted Arnoldi method [$O(k^3 + Nk^2 + N\#Ek)$] [Chen '11a]
2. high-dimensional k-means.

Overview

Problem : When N and/or k become too large, there are two main bottlenecks in the algorithm :

1. The partial eigendecomposition of the (sparse) Laplacian with restarted Arnoldi method [$O(k^3 + Nk^2 + N\#Ek)$] [Chen '11a]
2. high-dimensional k-means.

Goal : Circumvent bottleneck #1.

Overview

Problem : When N and/or k become too large, there are two main bottlenecks in the algorithm :

1. The partial eigendecomposition of the (sparse) Laplacian with restarted Arnoldi method [$O(k^3 + Nk^2 + N\#Ek)$] [Chen '11a]
2. high-dimensional k-means.

Goal : Circumvent bottleneck #1.

Scheme of existing lines of work :

1. approximate U_k : $\tilde{\mathrm{U}}_k$ via power methods [$O(pk\#E)$] [Lin '10, Boutsidis '15] or Nystrom-type methods [$O(n^3 + nN)$] [Fowlkes '04, Wang '09, Chen '11b, . . .].
2. compute approximate feature vectors $\tilde{\boldsymbol{f}}_i = \tilde{\mathrm{U}}_k^\top \boldsymbol{\delta}_i \simeq \boldsymbol{f}_i$.
3. compute approximate spectral distance $\tilde{D}_{ij} = \|\tilde{\boldsymbol{f}}_i - \tilde{\boldsymbol{f}}_j\| \simeq D_{ij}$.

Overview

Problem : When N and/or k become too large, there are two main bottlenecks in the algorithm :

1. The partial eigendecomposition of the (sparse) Laplacian with restarted Arnoldi method [$O(k^3 + Nk^2 + N\#Ek)$] [Chen '11a]
2. high-dimensional k-means.

Goal : Circumvent bottleneck #1.

Scheme of existing lines of work :

1. approximate U_k : $\tilde{\mathrm{U}}_k$ via power methods [$O(pk\#E)$] [Lin '10, Boutsidis '15] or Nystrom-type methods [$O(n^3 + nN)$] [Fowlkes '04, Wang '09, Chen '11b, . . .].
2. compute approximate feature vectors $\tilde{\boldsymbol{f}}_i = \tilde{\mathrm{U}}_k^\top \boldsymbol{\delta}_i \simeq \boldsymbol{f}_i$.
3. compute approximate spectral distance $\tilde{D}_{ij} = \|\tilde{\boldsymbol{f}}_i - \tilde{\boldsymbol{f}}_j\| \simeq D_{ij}$.

Contribution : directly estimate D_{ij} without approximating U_k in $O(m\#E \log N)$.

Filtering random graph signals for clustering

What's graph signal filtering ? [Shuman '13]

$$\boxed{L = S - W = U \Lambda U^\top}$$

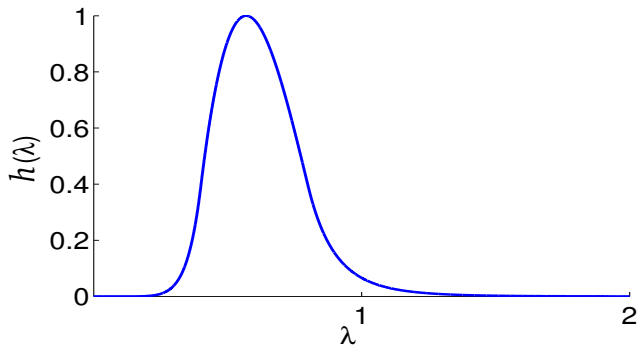
- U is the Fourier basis of the graph
- the Fourier transform of a signal f reads : $\boxed{\hat{f} = U^\top f}$
- $\Lambda = \mathrm{Diag}(\lambda_1, \lambda_2, \cdots, \lambda_N)$ the spectrum

What's graph signal filtering ? [Shuman '13]

$$\boxed{\mathsf{L} = \mathsf{S} - \mathsf{W} = \mathsf{U}\Lambda\mathsf{U}^\top}$$

- U is the Fourier basis of the graph
- the Fourier transform of a signal f reads : $\boxed{\hat{f} = \mathsf{U}^\top f}$
- $\Lambda = \mathrm{Diag}(\lambda_1, \lambda_2, \cdots, \lambda_N)$ the spectrum

Given a filter function h defined in the Fourier space.

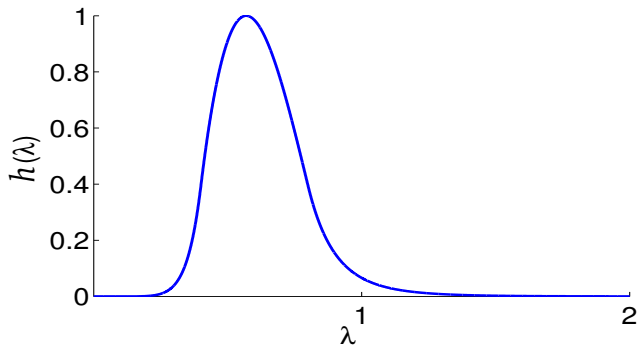


What's graph signal filtering ? [Shuman '13]

$$\boxed{\mathrm{L} = \mathrm{S} - \mathrm{W} = \mathrm{U}\Lambda\mathrm{U}^\top}$$

- U is the Fourier basis of the graph
- the Fourier transform of a signal f reads : $\boxed{\hat{f} = \mathrm{U}^\top f}$
- $\Lambda = \mathrm{Diag}(\lambda_1, \lambda_2, \cdots, \lambda_N)$ the spectrum

Given a filter function h defined in the Fourier space.



In the node space, the signal f filtered by h reads :

$$\boxed{f^h = \mathrm{U}\, h(\Lambda)\, \mathrm{U}^\top\, f = \mathrm{H}f}$$

So where's the link with clustering ?

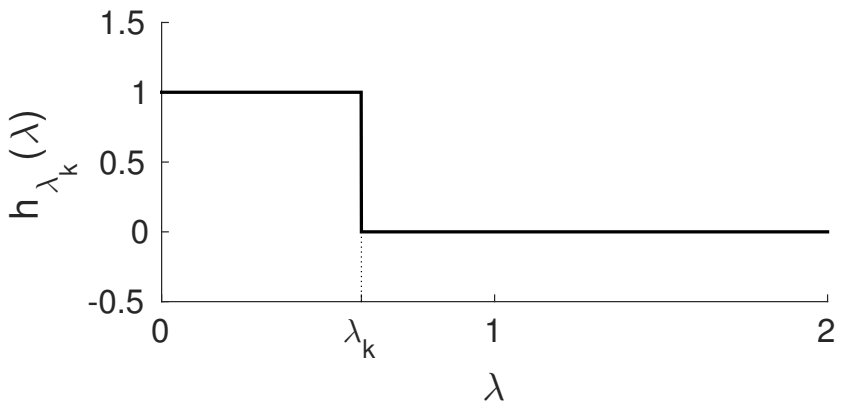
Remember : the classical spectral clustering algorithm

Given the N-node graph $\mathcal{G}$ of adjacency matrix W :

1. Compute L's first k eigenvectors : $\mathrm{U}_k = (\boldsymbol{u}_1|\boldsymbol{u}_2|\cdots|\boldsymbol{u}_k)$.
2. Consider each node i as a point in $\mathbb{R}^k$: $\boldsymbol{f}_i = \mathrm{U}_k^\top \boldsymbol{\delta}_i$.
3. Run k-means with $D_{ij} = ||\boldsymbol{f}_i - \boldsymbol{f}_j||$ and obtain k clusters.

Our goal : Estimate D_{ij} without computing exactly U_k.

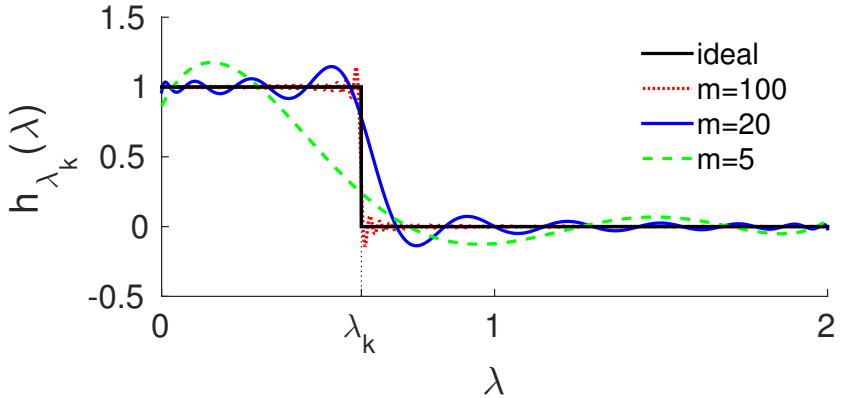
$$
\begin{aligned}
D_{ij} &= \left\| \mathrm{U}_k^\top (\boldsymbol{\delta}_i - \boldsymbol{\delta}_j) \right\| \\
&= \left\| \mathrm{U}_k^\top \boldsymbol{\delta}_{ij} \right\| \\
&= \left\| \mathrm{U}_k \mathrm{U}_k^\top \boldsymbol{\delta}_{ij} \right\| \\
&= \left\| \mathrm{U}\, h_{\lambda_k}(\Lambda)\, \mathrm{U}^\top \boldsymbol{\delta}_{ij} \right\| \\
&= \left\| \mathrm{H}_{\lambda_k} \boldsymbol{\delta}_{ij} \right\| .
\end{aligned}
$$



Fast filtering [Hammond, ACHA '11]

In practice, we use a poly approx of order m of h_{λ_k} :

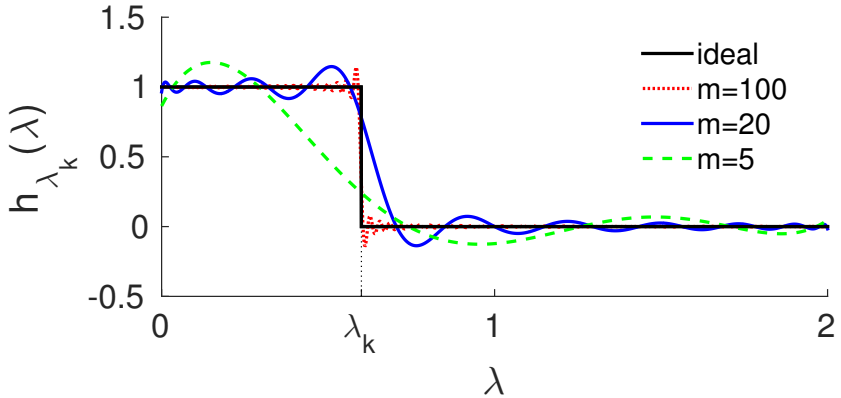
$$\tilde{h}_{\lambda_k} = \sum_{l=1}^{m} \alpha_l \lambda^l \simeq h_{\lambda_k}.$$



Fast filtering [Hammond, ACHA '11]

In practice, we use a poly approx of order m of h_{λ_k} :

$$\tilde{h}_{\lambda_k} = \sum_{l=1}^{m} \alpha_l \lambda^l \simeq h_{\lambda_k}.$$



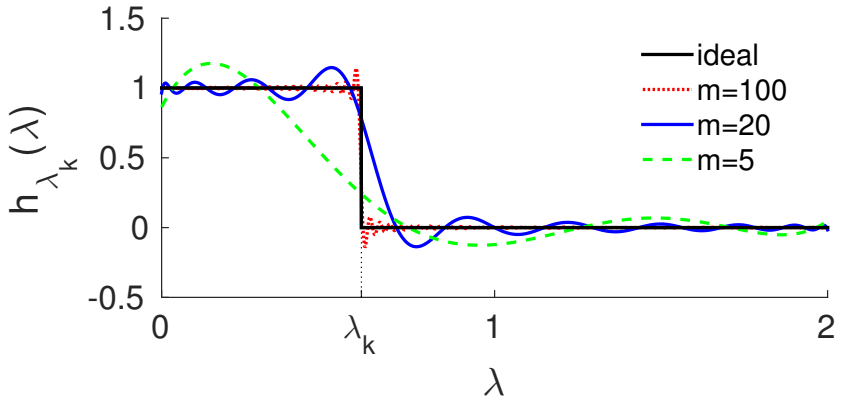
Indeed, in this case, filtering a signal $\boldsymbol{f}$ reads :

$$\mathrm{H}_{\lambda_k}\boldsymbol{f} \simeq \tilde{\mathrm{H}}_{\lambda_k}\boldsymbol{f} = \mathrm{U}\tilde{h}_{\lambda_k}(\Lambda)\mathrm{U}^\top \boldsymbol{f} = \mathrm{U}\sum_{l=1}^{m} \alpha_l \Lambda^l \mathrm{U}^\top \boldsymbol{f} = \sum_{l=1}^{m} \alpha_l \mathrm{L}^l f$$

Fast filtering [Hammond, ACHA '11]

In practice, we use a poly approx of order m of h_{λ_k} :

$$\tilde{h}_{\lambda_k} = \sum_{l=1}^{m} \alpha_l \lambda^l \simeq h_{\lambda_k}.$$



Indeed, in this case, filtering a signal $\boldsymbol{f}$ reads :

$$\mathrm{H}_{\lambda_k}\boldsymbol{f} \simeq \tilde{\mathrm{H}}_{\lambda_k}\boldsymbol{f} = \mathrm{U}\tilde{h}_{\lambda_k}(\Lambda)\mathrm{U}^\top \boldsymbol{f} = \mathrm{U}\sum_{l=1}^{m}\alpha_l \Lambda^l \mathrm{U}^\top \boldsymbol{f} = \sum_{l=1}^{m}\alpha_l \mathrm{L}^l f$$

- Does not require to compute U_k : no partial diagonalisation.
- Only involves matrix-vector multiplications [costs $O(m\#E)$].

Main idea

The spectral distance reads : $\boxed{D_{ij} = \|\mathrm{H}_{\lambda_k}\boldsymbol{\delta}_{ij}\| = \lim_{m\to\infty}\left\|\tilde{\mathrm{H}}_{\lambda_k}\boldsymbol{\delta}_{ij}\right\|}$

Main idea

The spectral distance reads : $\boxed{D_{ij} = \|\mathrm{H}_{\lambda_k}\boldsymbol{\delta}_{ij}\| = \lim_{m\to\infty}\left\|\tilde{\mathrm{H}}_{\lambda_k}\boldsymbol{\delta}_{ij}\right\|}$

Let $\mathrm{R} = (\boldsymbol{r}_1|\boldsymbol{r}_2|\cdots|\boldsymbol{r}_\eta) \in \mathbb{R}^{N\times\eta}$ be a random Gaussian matrix, i.e. a collection of η random graph signals, with 0 mean and var. $1/\eta$.

$$\boxed{\text{We define } \tilde{\boldsymbol{f}}_i = (\tilde{\mathrm{H}}_{\lambda_k}\mathrm{R})^\top\boldsymbol{\delta}_i \in \mathbb{R}^\eta \text{ and } \tilde{D}_{ij} = \left\|\tilde{\boldsymbol{f}}_i - \tilde{\boldsymbol{f}}_j\right\|}$$

Main idea

The spectral distance reads : $\boxed{D_{ij} = \|\mathrm{H}_{\lambda_k}\boldsymbol{\delta}_{ij}\| = \lim_{m\to\infty}\left\|\tilde{\mathrm{H}}_{\lambda_k}\boldsymbol{\delta}_{ij}\right\|}$

Let $\mathrm{R} = (\boldsymbol{r}_1|\boldsymbol{r}_2|\cdots|\boldsymbol{r}_\eta) \in \mathbb{R}^{N\times\eta}$ be a random Gaussian matrix, i.e. a collection of η random graph signals, with 0 mean and var. $1/\eta$.

$$\boxed{\text{We define } \tilde{\boldsymbol{f}}_i = (\tilde{\mathrm{H}}_{\lambda_k}\mathrm{R})^\top\boldsymbol{\delta}_i \in \mathbb{R}^\eta \text{ and } \tilde{D}_{ij} = \left\|\tilde{\boldsymbol{f}}_i - \tilde{\boldsymbol{f}}_j\right\|}$$

Norm conservation theorem (case of infinite m)

Let $\epsilon > 0$, if $\eta > \eta_0 \sim \frac{\log N}{\epsilon^2}$, then, with proba $> 1 - 1/N$, we have :

$$\forall (i,j) \in [1,N]^2 \qquad (1-\epsilon)D_{ij} \leq \tilde{D}_{ij} \leq (1+\epsilon)D_{ij}.$$

Main idea

The spectral distance reads : $\boxed{D_{ij} = \|\mathrm{H}_{\lambda_k}\boldsymbol{\delta}_{ij}\| = \lim_{m\to\infty}\left\|\tilde{\mathrm{H}}_{\lambda_k}\boldsymbol{\delta}_{ij}\right\|}$

Let $\mathrm{R} = (\boldsymbol{r}_1|\boldsymbol{r}_2|\cdots|\boldsymbol{r}_\eta) \in \mathbb{R}^{N\times\eta}$ be a random Gaussian matrix, i.e. a collection of η random graph signals, with 0 mean and var. $1/\eta$.

$$\boxed{\text{We define } \tilde{\boldsymbol{f}}_i = (\tilde{\mathrm{H}}_{\lambda_k}\mathrm{R})^\top\boldsymbol{\delta}_i \in \mathbb{R}^\eta \text{ and } \tilde{D}_{ij} = \left\|\tilde{\boldsymbol{f}}_i - \tilde{\boldsymbol{f}}_j\right\|}$$

Norm conservation theorem (case of infinite m)

Let $\epsilon > 0$, if $\eta > \eta_0 \sim \frac{\log N}{\epsilon^2}$, then, with proba $> 1 - 1/N$, we have :

$$\forall (i,j) \in [1,N]^2 \qquad (1-\epsilon)D_{ij} \leq \tilde{D}_{ij} \leq (1+\epsilon)D_{ij}.$$

Consequence : to estimate D_{ij} with no partial diagonalisation of L, fast filter only $\eta \sim \log N$ random signals !

How to quickly estimate λ_k, the sole unknown of the fast filtering operation ?

Goal : given a SDP L, estimate its k-th eigenvalue as fast as possible.

We use eigencount techniques [Napoli '13] (also based on polynomial filtering of random vectors !) :

- given the interval $[0, b]$, get an approximation of the number of enclosed eigenvalues.
- And find λ_k by dichotomy on b.

Accelerated spectral algorithm

Given the N-node graph $\mathcal{G}$ of adjacency matrix W :

Inputs : k, polynomial order m, # random signals $\eta \sim \log N$.

1. Estimate λ_k, the k-th eigenvalue of L.
2. Generate η random graph signals in matrix $\mathrm{R} \in \mathbb{R}^{N \times \eta}$.
3. Filter them with $\tilde{\mathrm{H}}_{\lambda_k}$ and treat each node i as a point in $\mathbb{R}^{\eta}$:

$$\tilde{\boldsymbol{f}}_i^{\top} = \boldsymbol{\delta}_i^{\top} \tilde{\mathrm{H}}_{\lambda_k} \mathrm{R}.$$

4. Run k-means with the Euclidean distance : $\tilde{D}_{ij} = ||\tilde{\boldsymbol{f}}_i - \tilde{\boldsymbol{f}}_j||$ and obtain k clusters.

Accelerated spectral algorithm

Given the N-node graph $\mathcal{G}$ of adjacency matrix W :

Inputs : k, polynomial order m, # random signals $\eta \sim \log N$.

1. Estimate λ_k, the k-th eigenvalue of L.
2. Generate η random graph signals in matrix $\mathrm{R} \in \mathbb{R}^{N \times \eta}$.
3. Filter them with $\tilde{\mathrm{H}}_{\lambda_k}$ and treat each node i as a point in $\mathbb{R}^{\eta}$:

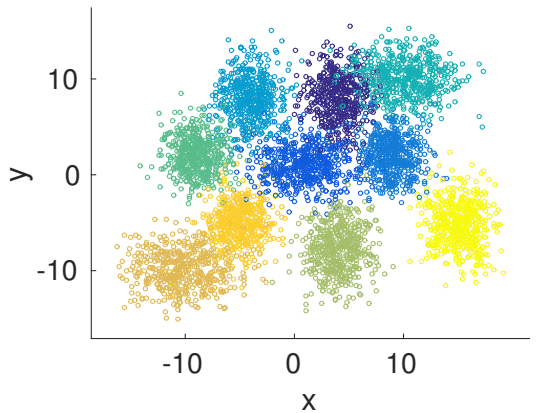
$$\tilde{\boldsymbol{f}}_i^{\top} = \boldsymbol{\delta}_i^{\top} \tilde{\mathrm{H}}_{\lambda_k} \mathrm{R}.$$

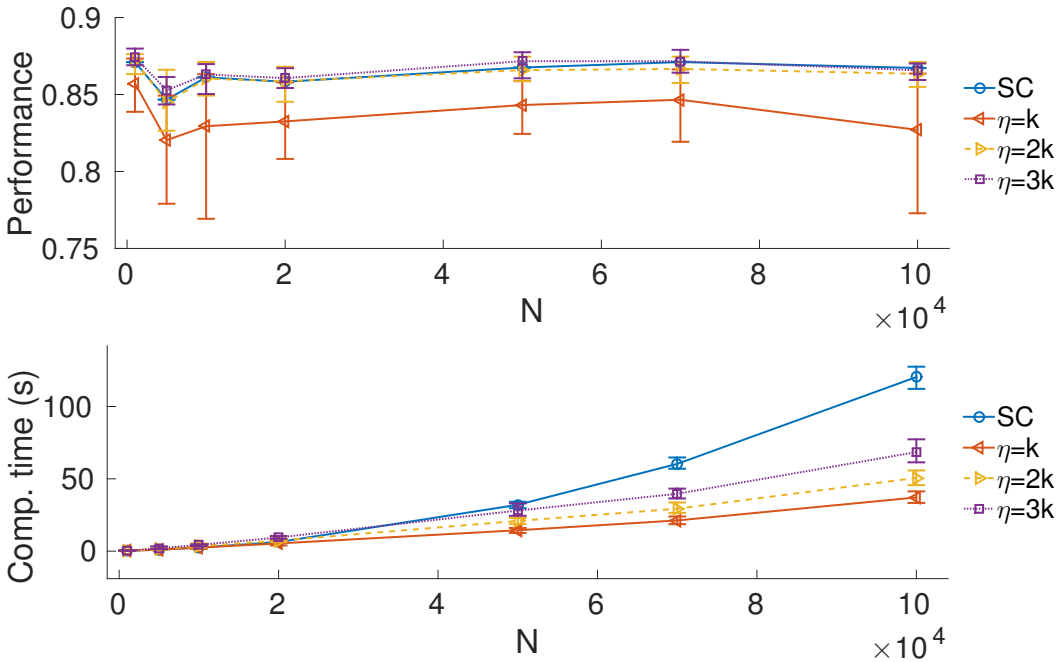
4. Run k-means with the Euclidean distance : $\tilde{D}_{ij} = ||\tilde{\boldsymbol{f}}_i - \tilde{\boldsymbol{f}}_j||$ and obtain k clusters.

Computing the features (steps 1 to 3) costs $O(m\#E \log N) \simeq O(mN \log N)$.

Toy experiment

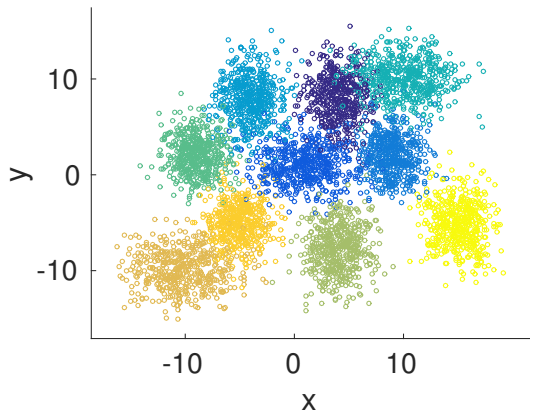
Gaussian data around 10 centroids :

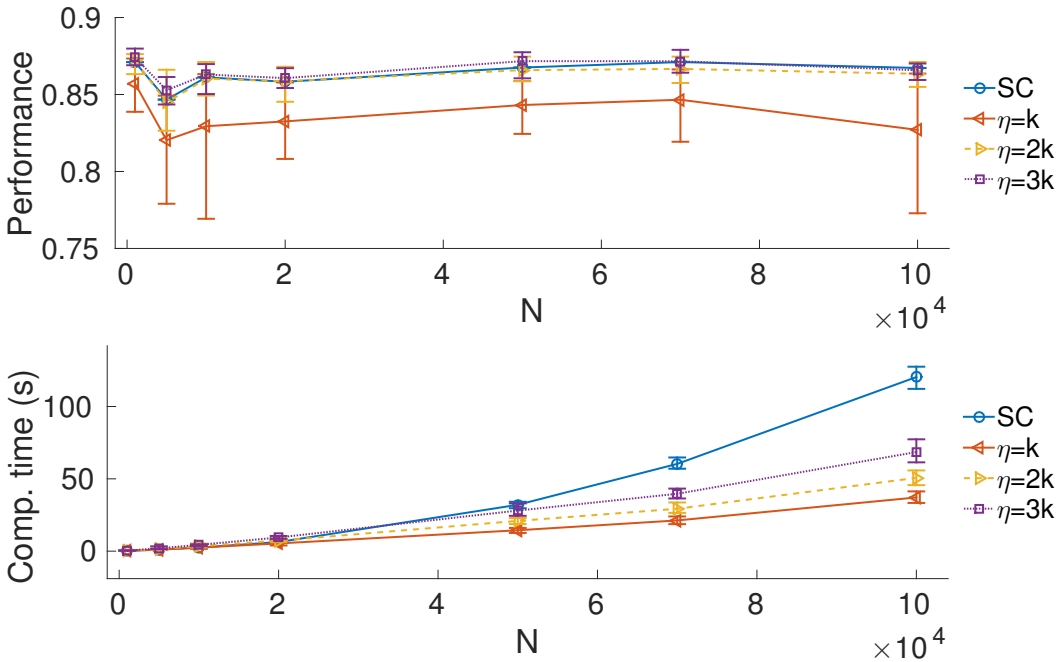




Toy experiment

Gaussian data around 10 centroids :





Example : $N = 10^5, k = 10$: filtering only $\eta = 20$ random signals gives same performance... 3x faster.

Main idea of this paper

Low-pass graph fast filtering of random signals : a way to by-pass the Laplacian's diagonalisation for learning tasks.

Ideas surrounding this paper : *compressive* spectral clustering [arXiv : Tremblay '16, Puy '15]

1. **Cluster indicator functions are k-bandlimited signals on the graph** : they can be recovered after appropriate sampling schemes ! 1/ Compute features, 2/ Sample a few nodes, 3/ Run low-dimensional k-means, 4/ Interpolate
2. **Norm conservation theorem for finite polynomial order m**
3. **Generalisation to the use of the normalized Laplacian** (but not the random walk Laplacian yet)

Perspectives and difficult questions

Two difficult questions (among others) :

1. Given a SDP matrix, how to estimate as fast as possible its k-th eigenvalue, and only that one ?
2. How to choose automatically the appropriate polynomial order m ?

Perspectives and difficult questions

Two difficult questions (among others) :

1. Given a SDP matrix, how to estimate as fast as possible its k-th eigenvalue, and only that one ?
2. How to choose automatically the appropriate polynomial order m ?

Perspectives

1. Rational filters instead of polynomial filters ? [Shi '15b]
2. How about if nodes are added one by one ?

References

[Ramasamy '15] *Compressive spectral embedding : sidestepping . . .* NIPS.
[Fortunato '10] *Community detection in graphs*, Physics Reports
[Newman '06] *Modularity and community structure in networks*, PNAS
[Schaub '12] *Markov dynamics as a zooming lens for multiscale . . .*, Plos One
[Krzakala '13] *Spectral redemption : clustering sparse networks*, PNAS
[Rosvall '08] *Maps of random walks on complex networks reveal . . .*, PLOS One
[Von Luxburg '06] *A tutorial on spectral clustering*, Statistics and Computing.
[Chen '11a] *Parallel spectral clustering in distributed systems*, IEEE TPAMI
[Lin '10] *Power iteration clustering*, ICML
[Boutsidis '15] *Spectral clustering via the power method - provably*, ICML
[Fowlkes '04] *Spectral grouping using the nystrom method*, IEEE TPAMI
[Wang '09] *Approximate spectral clustering*, AKDDM
[Chen '11b] *Large scale spectral clustering with landmark-based . . .*, CAI
[Shuman '13] *The emerging field of signal processing on graphs . . .*, SPMag
[Hammond '11] *Wavelets on graphs via spectral graph theory*, ACHA
[Napoli '13] *Efficient estimation of eigenvalue counts in an interval*, arXiv
[Tremblay '16] *Compressive spectral clustering*, arXiv, subm. to ICML
[Puy '15] *Random sampling of bandlimited signals . . .*, arXiv, subm. to ACHA
[Shi '15] *Infinite impulse response graph filters in wireless sensor networks*, SPL