



Signal Processing on Graphs

P. Vandergheynst
CIS - Center for
Intelligent Systems
EPFL

Peyresq
June 2023

Previous Episode

Eigen decomposition of Laplacian reveals structure of graph

Spectral Clustering: Exploit eigenvectors to identify communities

Ingredient: Smoothness of feature vectors (smooth partition signals)

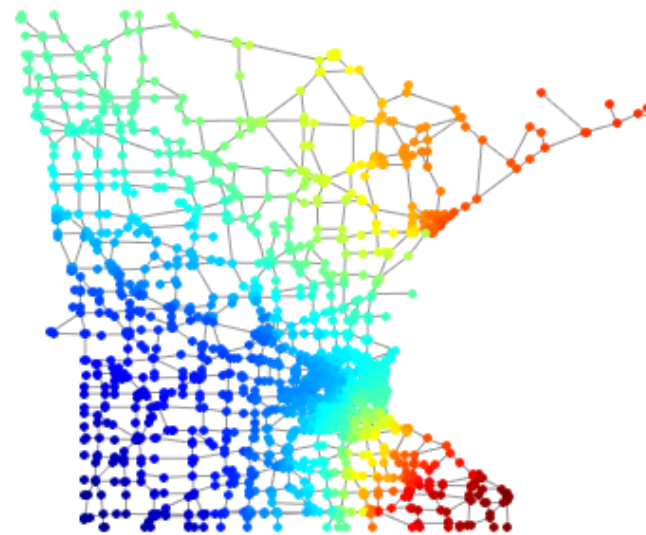
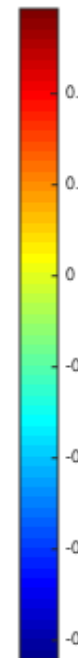
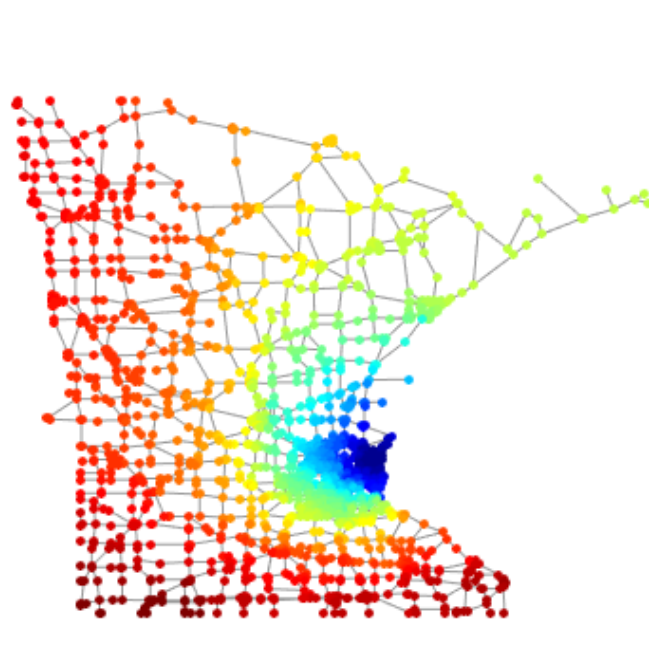
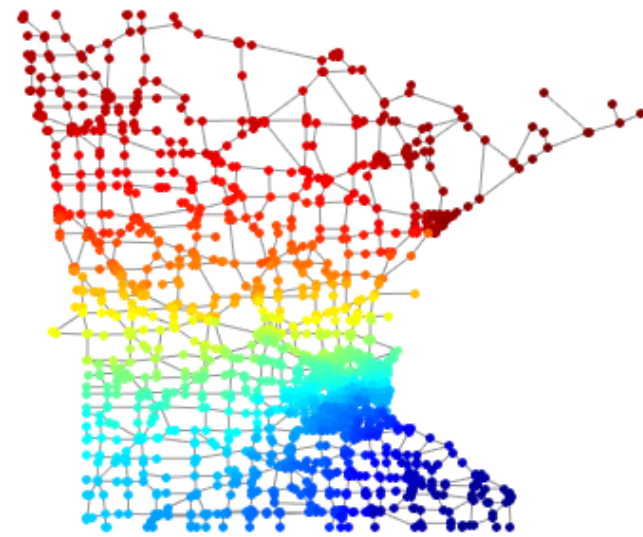
Eigenmaps: Exploit eigenvectors to compute low-dimensional embeddings

Ingredient: Smoothness of coordinate maps

Today

create models of smooth graph signals and
operators to manipulate them (GSP)

A Few Laplacian Eigenvectors



Functional calculus

It will be useful to manipulate functions of the Laplacian

$$f(\mathbf{L}), f : \mathbb{R} \mapsto \mathbb{R}$$

$$\mathbf{L}^k \mathbf{u}_\ell = \lambda_\ell^k \mathbf{u}_\ell \longrightarrow \text{polynomials}$$

Symmetric matrices admit a functional calculus

functional calculus for symmetric matrices

$$f(\mathbf{L}) = \sum_{\ell \in \mathcal{S}(\mathbf{L})} f(\lambda_\ell) \mathbf{u}_\ell \mathbf{u}_\ell^t$$

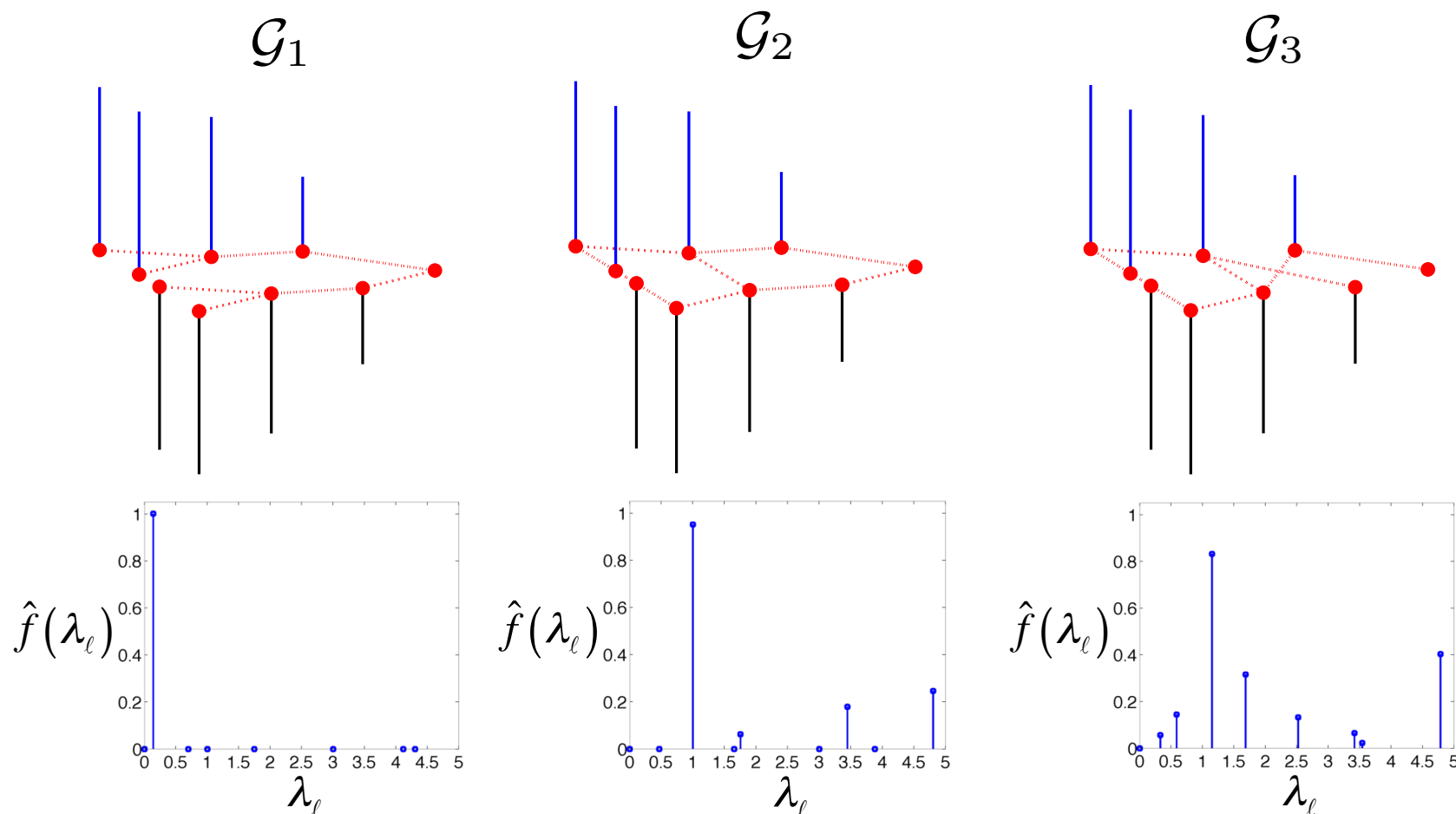
Use spectral theorem on powers, get to polynomials, ...

Graph Fourier Transform

GFT of a graph signal

$$f : V \rightarrow \mathbb{R} \quad \hat{f}[\ell] = u_\ell^T f \quad \hat{f}[\ell] = \sum_{i \in V} f[i] u_\ell[i]$$

$$f = \sum_{\ell} \hat{f}[\ell] u_\ell$$

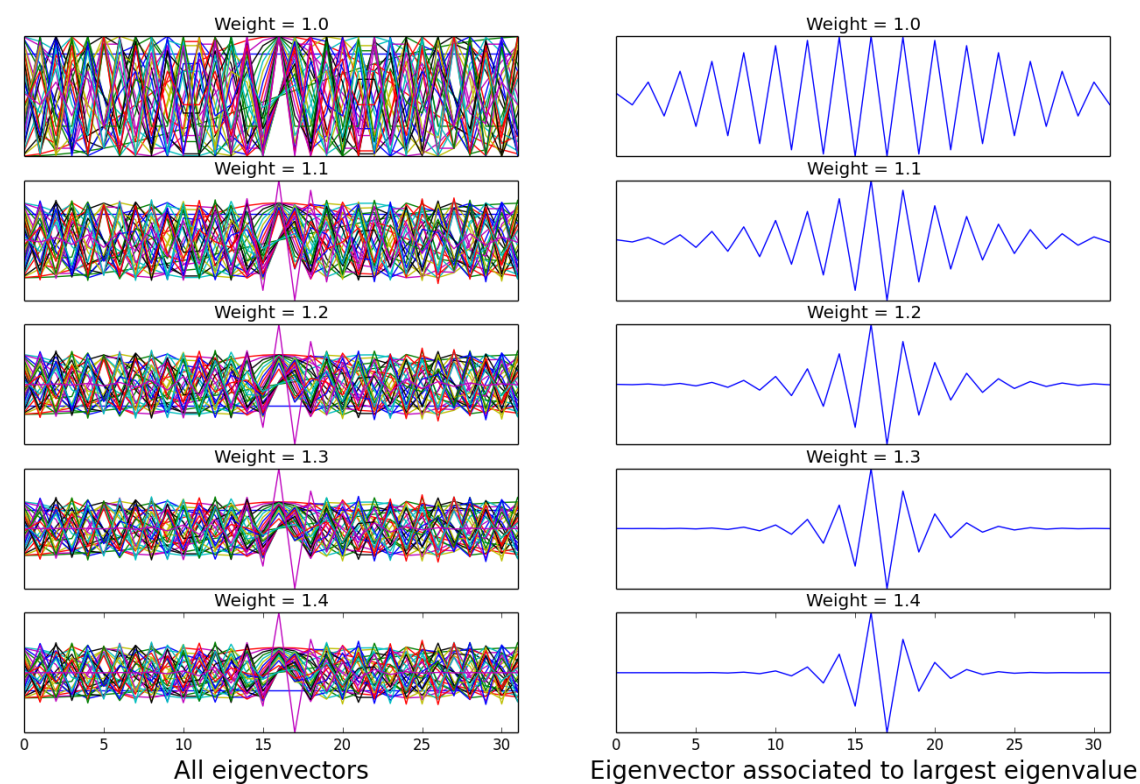


Graph Fourier Transform

$$\mu := \max_{\ell, i} |\langle \mathbf{u}_\ell, \delta_i \rangle| \in \left[\frac{1}{\sqrt{N}}, 1 \right]$$

Graph coherence

Optimal - Fourier case What does that mean ??



Eigenvectors of modified path graph

Example: Diffusion on Graphs

Consider the following « heat » diffusion model

$$\frac{\partial f}{\partial t} = -\mathbf{L}f \quad \frac{\partial}{\partial t} \hat{f}(\ell, t) = -\lambda_\ell \hat{f}(\ell, t) \quad \hat{f}(\ell, 0) := \hat{f}_0(\ell)$$

$$\hat{f}(\ell, t) = e^{-t\lambda_\ell} \hat{f}_0(\ell) \quad f = e^{-t\mathbf{L}} f_0 \quad \text{by spectral theorem}$$

Explicitly:

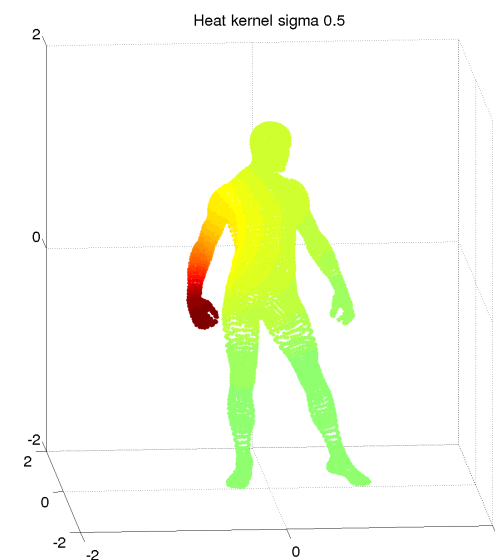
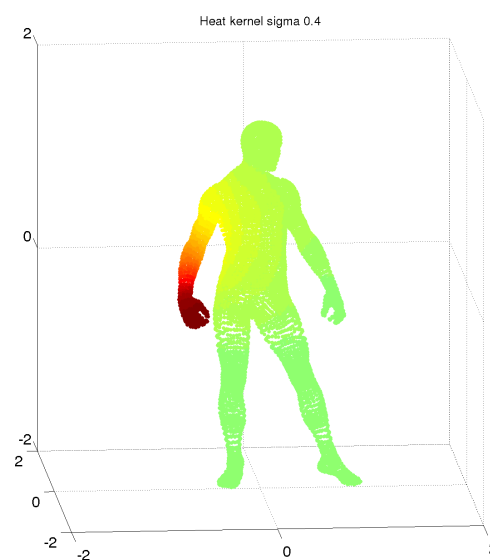
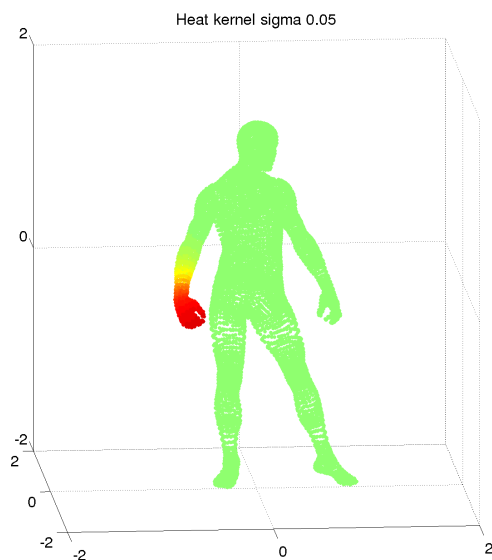
$$f(i) = \sum_{j \in V} \sum_{\ell} e^{-t\lambda_\ell} u_\ell(i) u_\ell(j) f_0(j)$$

$$e^{-t\mathcal{L}} = \sum_{\ell} e^{-t\lambda_\ell} \mathbf{u}_\ell \mathbf{u}_\ell^t = \sum_{\ell} e^{-t\lambda_\ell} u_\ell(i) \sum_{j \in V} u_\ell(j) f_0(j)$$

$$e^{-t\mathcal{L}}[i, j] = \sum_{\ell} e^{-t\lambda_\ell} u_\ell(i) u_\ell(j) = \sum_{\ell} e^{-t\lambda_\ell} \hat{f}_0(\ell) u_\ell(i)$$

Example: Diffusion on Graphs

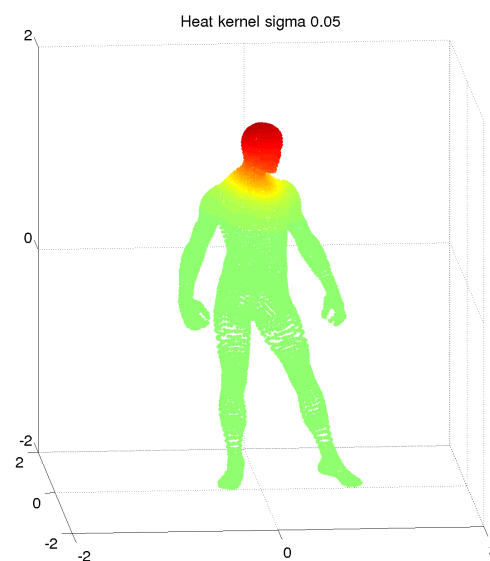
examples of heat kernel on graph



$$f_0(j) = \delta_k(j)$$

$$f(i) = \sum_{\ell} e^{-t\lambda_{\ell}} \hat{f}_0(\ell) u_{\ell}(i)$$

$$= \sum_{\ell} e^{-t\lambda_{\ell}} u_{\ell}(k) u_{\ell}(i)$$



Diffusion = Gradient Flow

Dirichlet semi-norm characterises smoothness

$$\mathcal{D}(f) = \frac{1}{2} \|\mathbf{S}^T f\|_2^2$$

Gradient flow: $\delta_u \mathcal{D}(f) = \langle u, \nabla \mathcal{D}(f) \rangle$
 $= \langle u, \mathbf{L} f \rangle$

$$\frac{\partial f}{\partial t} = -\mathbf{L} f$$

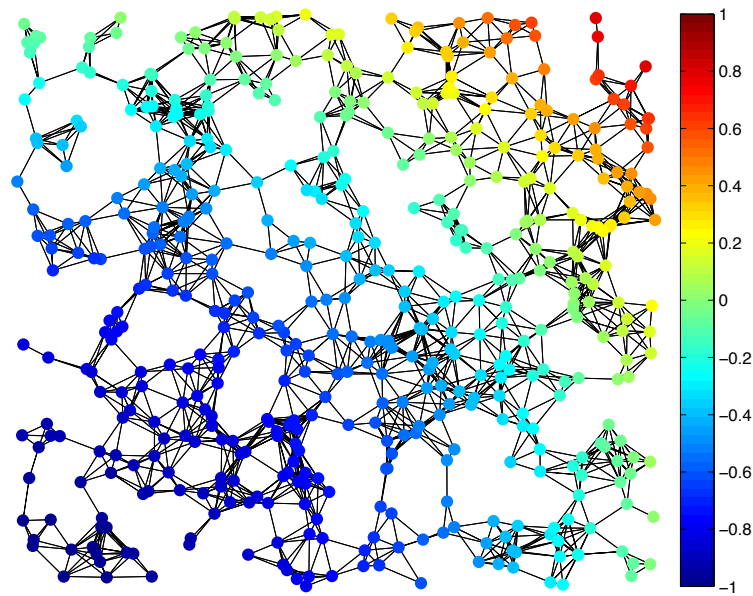
Using our definition of gradient: $\nabla_u f = \{S^T f(u, v), \forall v \sim u\}$

Local variation $\|\nabla_u f\|_2 = \sqrt{\sum_{v \sim u} |S^T f(u, v)|^2}$

Total variation $|f|_{TV} = \sum_{u \in V} \|\nabla_u f\|_2 = \sum_{u \in V} \sqrt{\sum_{v \sim u} |S^T f(u, v)|^2}$

Simple De-Noising Example

Suppose a smooth signal f on a graph



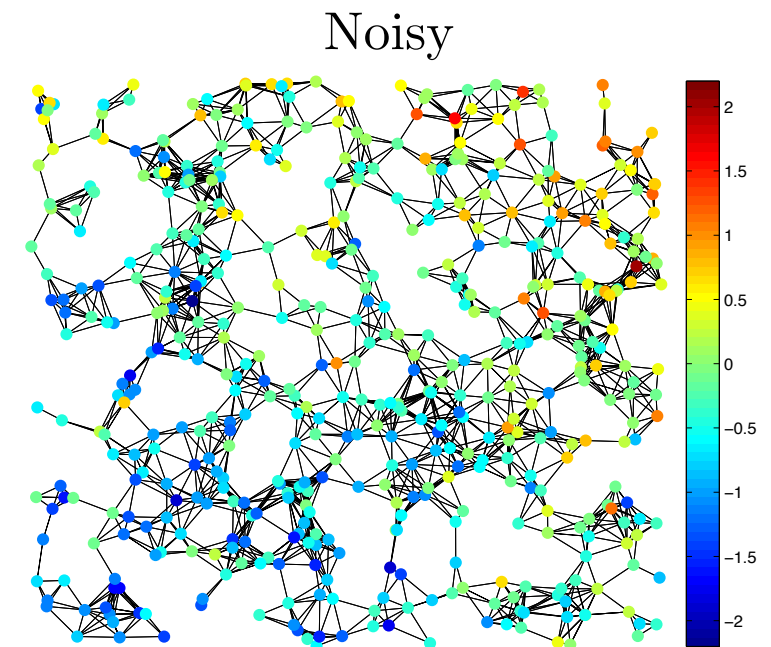
Original

$$\|S^T f\|_2^2 \leq M \Leftrightarrow f^t \mathbf{L} f \leq M$$

$$|\hat{f}(\ell)| \leq \frac{\sqrt{M}}{\sqrt{\lambda_\ell}}$$

But you observe only a noisy version y

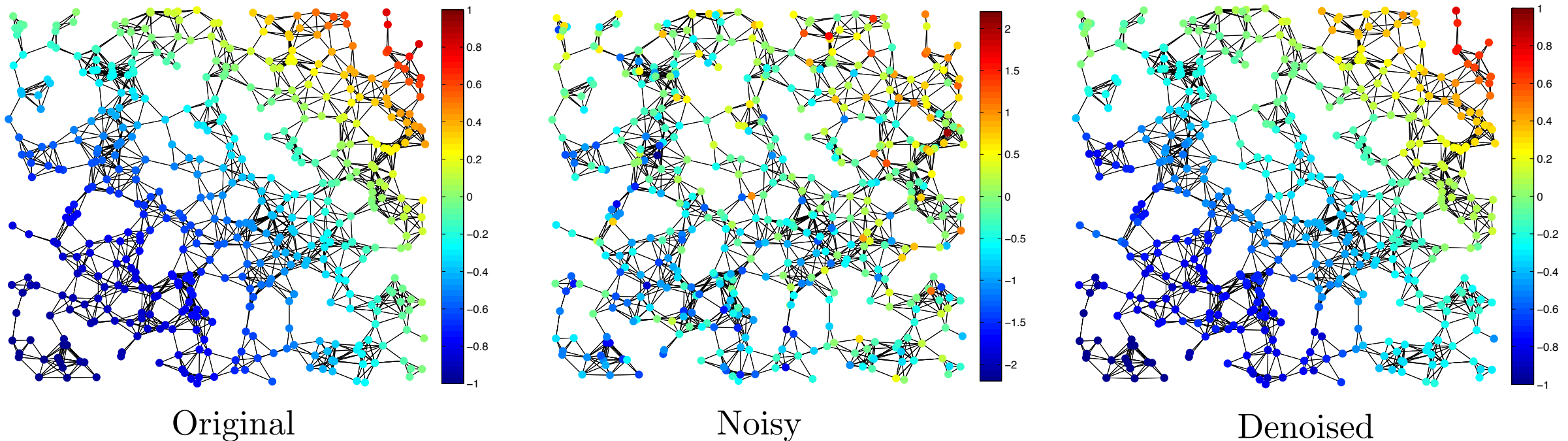
$$y(i) = f(i) + n(i)$$



Noisy

Simple De-Noising Example

$$\operatorname{argmin}_f \left\{ \frac{\tau}{2} \|f - y\|_2^2 + f^T \mathbf{L}^r f \right\}$$



$$\operatorname{argmin}_f \left\{ \frac{\tau}{2} \|f - y\|_2^2 + f^T \mathbf{L}^r f \right\} \quad \Rightarrow \quad \mathbf{L}^r f_* + \frac{\tau}{2} (f_* - y) = 0$$

$$\Rightarrow \quad \hat{f}_*(\ell) = \frac{\tau}{\tau + 2\lambda_\ell^r} \hat{y}(\ell) \quad \text{“Low pass” filtering !}$$

$$\text{Filtering: } \hat{f}_{out}(\lambda_\ell) = \hat{f}_{in}(\lambda_\ell) \hat{h}(\lambda_\ell) \quad f_{out}(i) = \sum_{\ell=0}^{N-1} \hat{f}_{in}(\lambda_\ell) \hat{h}(\lambda_\ell) u_\ell(i)$$

Graph Filters

A Graph Filter is an operator acting on graph signals

It is represented by a function of the Laplacian: $g(\mathbf{L}) \in \mathbb{R}^{N \times N}$

$$f_{\text{out}} = g(\mathbf{L})f_{\text{in}}$$

Via functional calculus or an explicit calculation: $\widehat{f_{\text{out}}}(\lambda_\ell) = g(\lambda_\ell)\widehat{f_{\text{in}}}(\lambda_\ell)$

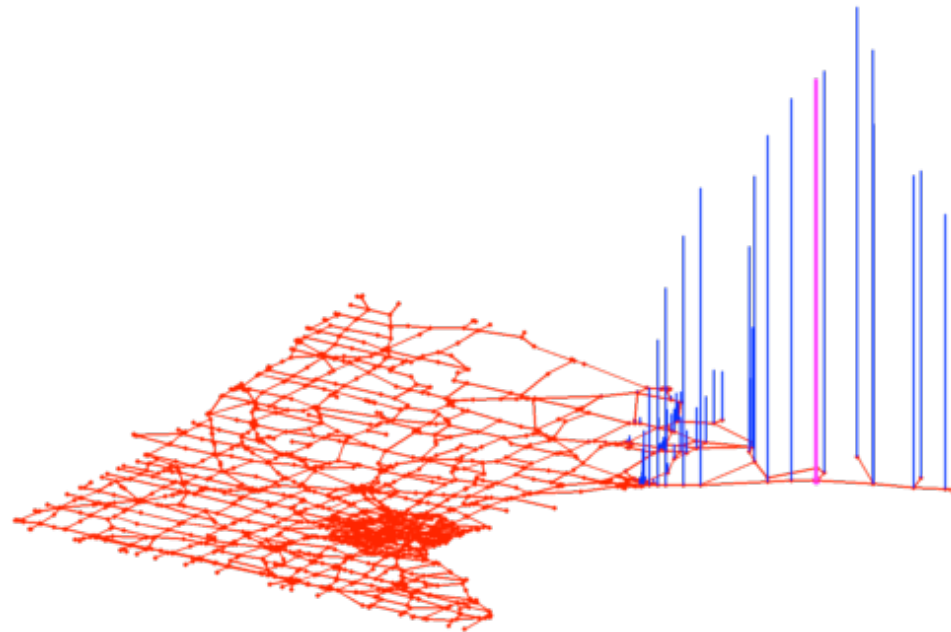
A graph filter reshapes frequency content

This operation is sometimes called “convolution over graphs”

Graph Filters as Features

Filtering = Correlation with a localised pattern

$$\begin{aligned}
 (g(\mathbf{L})f)[i] &= \sum_{\ell} g(\lambda_{\ell}) \hat{f}(\lambda_{\ell}) u_{\ell}[i] \\
 &= \sum_j f[j] \boxed{\sum_{\ell} g(\lambda_{\ell}) u_{\ell}[i] u_{\ell}[j]} \\
 &= \langle T_i g, f \rangle \longrightarrow (T_i g) = g(\mathbf{L}) \delta_i
 \end{aligned}$$



Polynomial Localization

Given a spectral kernel g , construct the family of features:

$$\phi_n[m] = (T_n g)[m] \qquad \phi_n[m] = \sum_{\ell} g(\lambda_{\ell}) u_{\ell}[m] u_{\ell}[n]$$

Are these features localized ?

Polynomial Kernels are K -Localized

$$\widehat{p_K}(\lambda_{\ell}) = \sum_{k=0}^K a_k \lambda_{\ell}^k \qquad \text{if } d(i, n) > K, \text{ then } (T_i p_K)(n) = 0$$

Polynomial Localization

Given a spectral kernel g , construct the family of features:

$$\phi_n[m] = (T_n g)[m] \qquad \phi_n[m] = \sum_{\ell} g(\lambda_{\ell}) u_{\ell}[m] u_{\ell}[n]$$

Are these features localized ?

Suppose the GFT of the kernel is smooth enough ($K+1$ different.)

Construct an order K polynomial approximation:

$$\phi'_n[m] = \langle \delta_m, P_K(\mathbf{L}) \delta_n \rangle \qquad \text{Exactly localized in a } K\text{-ball around } n$$

$$\phi_n[m] = \langle \delta_m, g(\mathbf{L}) \delta_n \rangle$$



Should be well localized within K -ball around n !

Polynomial Localization - Extended

f is $(K+1)$ -times differentiable:

$$\inf_{q_K} \{ \|f - q_K\|_\infty \} \leq \frac{\left[\frac{b-a}{2}\right]^{K+1}}{(K+1)! 2^K} \|f^{(K+1)}\|_\infty$$

Let $K_{in} := d(i, n) - 1$

$$|(T_i g)(n)| \leq \sqrt{N} \inf_{\widehat{p_{K_{in}}}} \left\{ \sup_{\lambda \in [0, \lambda_{\max}]} |\hat{g}(\lambda) - \widehat{p_{K_{in}}}(\lambda)| \right\} = \sqrt{N} \inf_{\widehat{p_{K_{in}}}} \{ \|\hat{g} - \widehat{p_{K_{in}}}\|_\infty \}$$

Regular Kernels are Localized

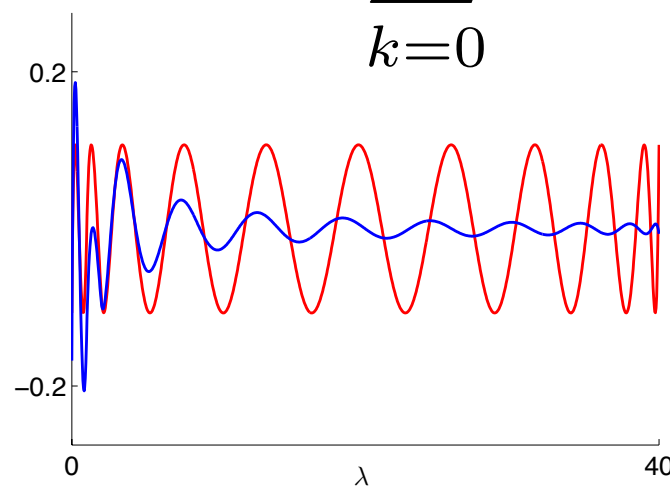
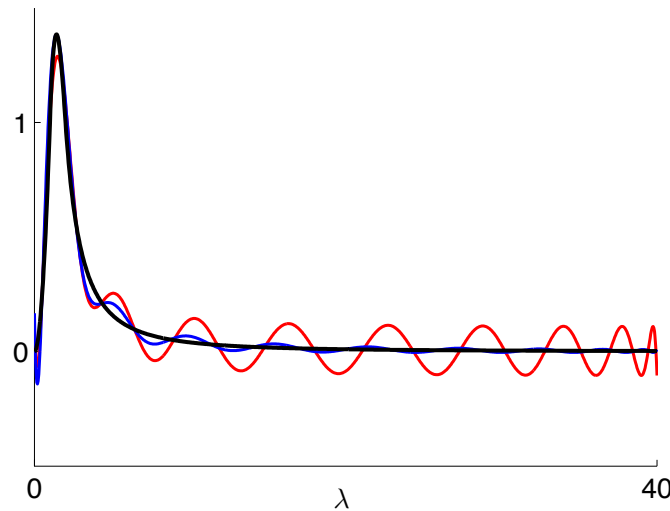
If the kernel is $d(i, n)$ -times differentiable:

$$|(T_i g)(n)| \leq \left[\frac{2\sqrt{N}}{d_{in}!} \left(\frac{\lambda_{\max}}{4} \right)^{d_{in}} \sup_{\lambda \in [0, \lambda_{\max}]} |\hat{g}^{(d_{in})}(\lambda)| \right]$$

Remark on Implementation

Not necessary to compute spectral decomposition

Polynomial approximation : $g(x) \simeq \sum_{k=0}^{K-1} a_k p_k(x)$ ex: Chebyshev, minimax



Then graph filter expressed with powers of Laplacian:

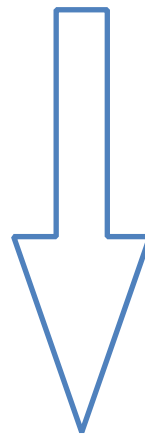
$$g(\mathbf{L}) \simeq \sum_{k=0}^{K-1} a_k \mathbf{L}^k$$

And use sparsity of Laplacian in an iterative way

Remark on Implementation

Chebyshev:
$$g_{\Theta}(\mathbf{L}) = \sum_{k=0}^{K-1} \Theta_k T_k(\bar{\mathbf{L}}) \quad \bar{\mathbf{L}} = 2\mathbf{L}/\lambda_{\max} - \mathbf{I}_N$$

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x), \quad T_0(x) = 1, \quad T_1(x) = x$$



Recursive implementation of filter by message passing
OR sparse matrix-vector multiplication



Designing and Processing Graph Features with GSP

Spectral Graph Wavelets

 Hammond et al., Wavelets on graphs via spectral graph theory, ACHA, 2011
Generalized translation

► Classical setting:

$$(T_s g)(t) = g(t - s) = \int_{\mathbb{R}} \hat{g}(\xi) e^{-2\pi i \xi s} e^{2\pi i \xi t} d\xi$$

► Graph setting:

$$(T_n g)(i) := \sum_{\ell=0}^{N-1} \hat{g}(\lambda_\ell) u_\ell^*(n) u_\ell(i)$$

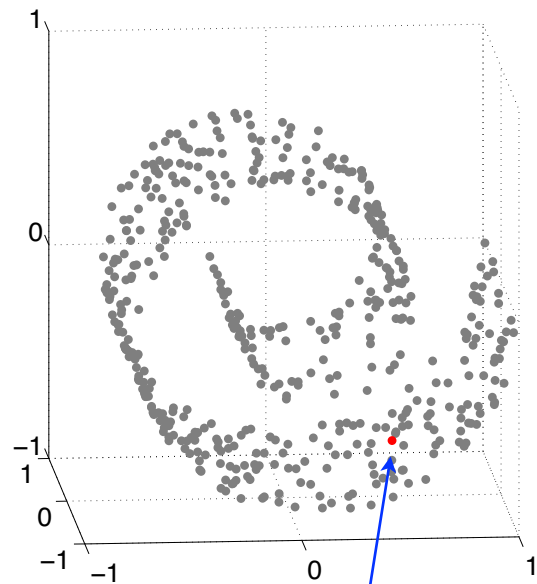
• Generalized dilation:

$$\widehat{\mathcal{D}_s g}(\lambda) = \hat{g}(s\lambda)$$

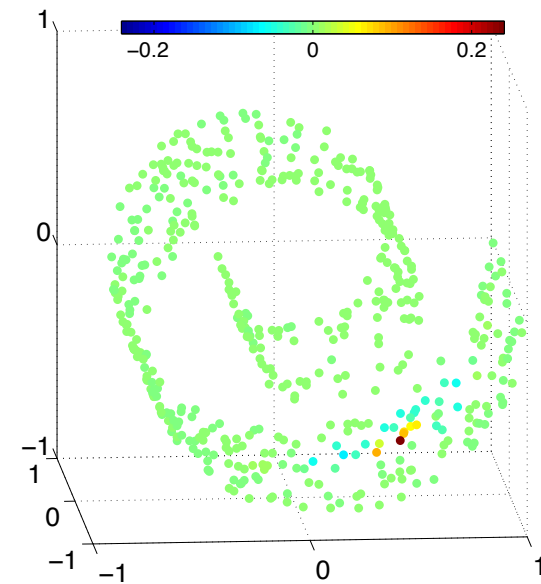
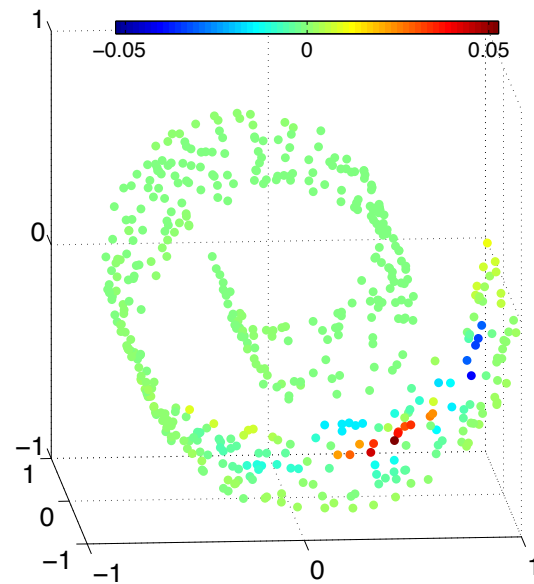
• Spectral graph wavelet at scale s , centered at vertex n :

$$\psi_{s,n}(i) := (T_n \mathcal{D}_s g)(i) = \sum_{\ell=0}^{N-1} \hat{g}(s\lambda_\ell) u_\ell^*(n) u_\ell(i)$$

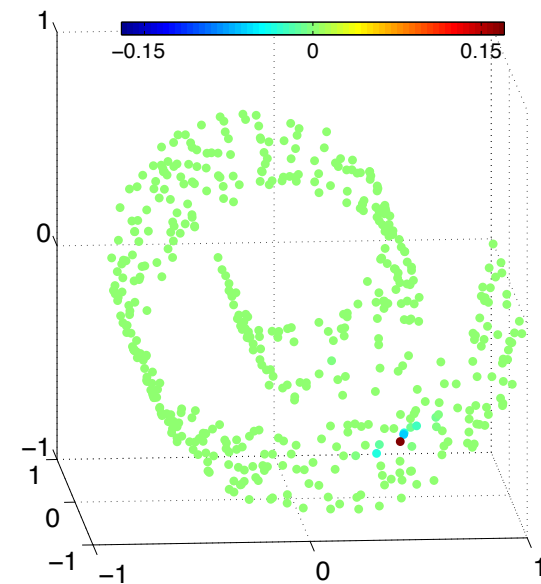
Scaling & Localization



$$\psi_{t,i}(j)$$



decreasing scale



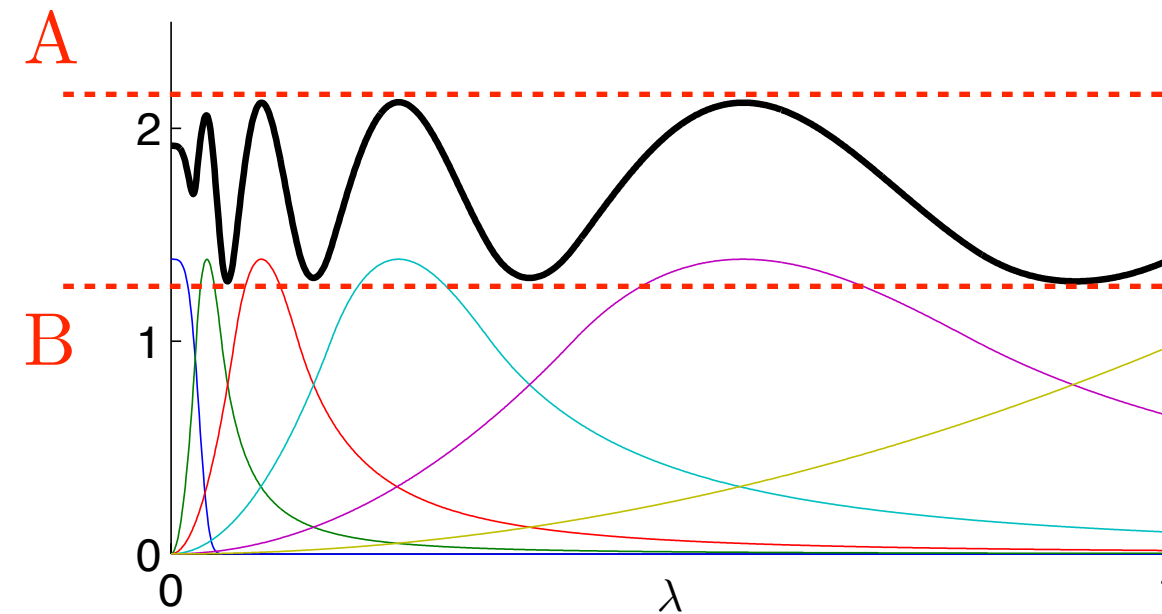
Frames

$\exists A, B > 0, \exists \hat{h} : \mathbb{R}^+ \mapsto \mathbb{R}$ i.e scaling function

$$0 < A \leq \hat{h}(u)^2 + \sum_s \hat{g}(t_s u)^2 \leq B < +\infty$$

$\swarrow$
scaling function
 $\swarrow$
wavelets

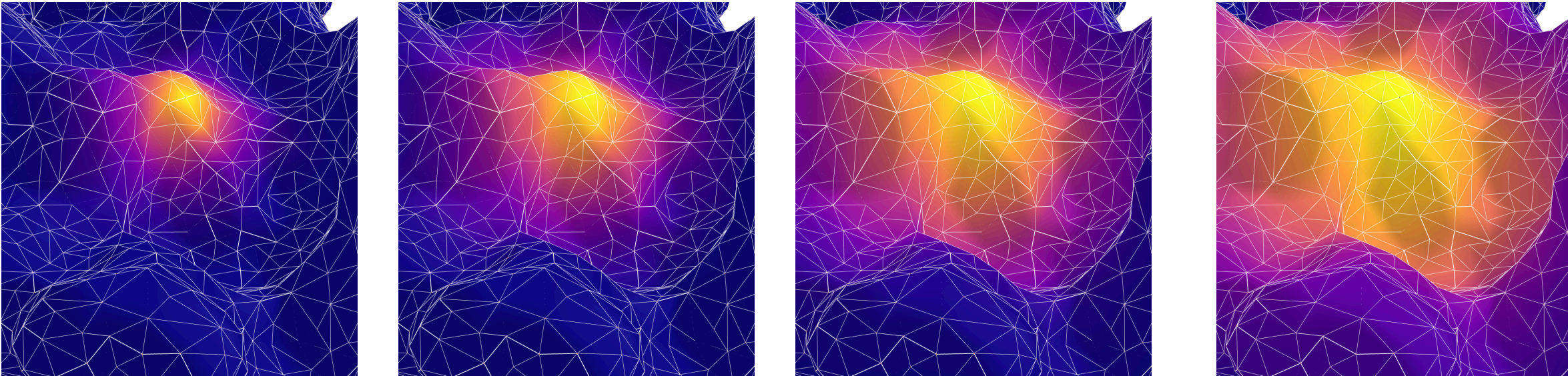
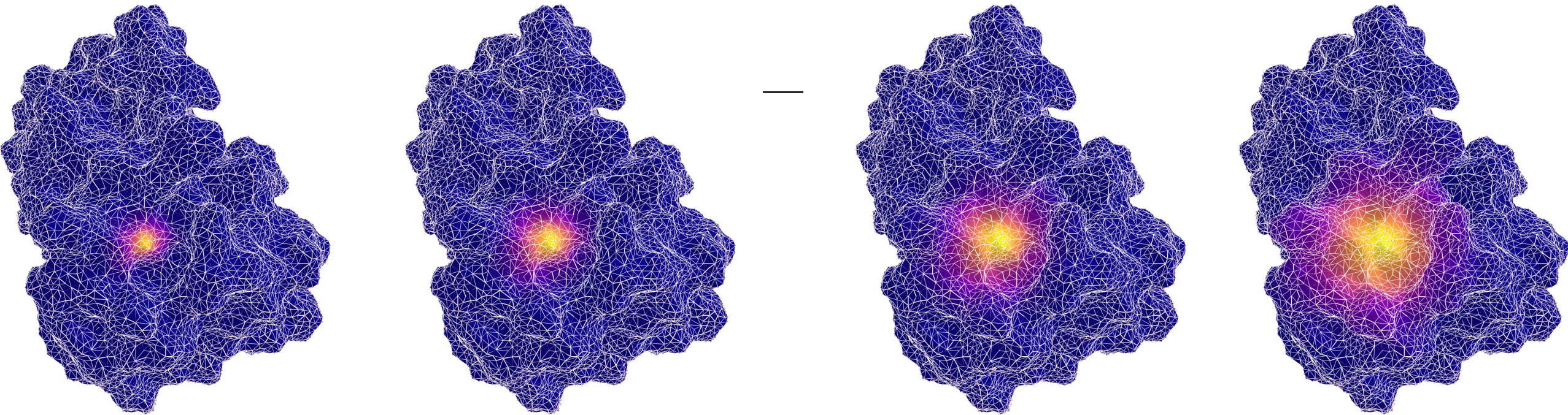
$$\phi_n = W_h \delta_n = h(L) \delta_n$$



A simple way to get a tight frame:

$$\hat{\gamma}(\lambda_\ell) = \int_{1/2}^1 \frac{dt}{t} \hat{g}(t\lambda_\ell)^2 \Rightarrow \tilde{\hat{g}}(\lambda_\ell) = \sqrt{\hat{\gamma}(\lambda_\ell) - \hat{\gamma}(2\lambda_\ell)}$$

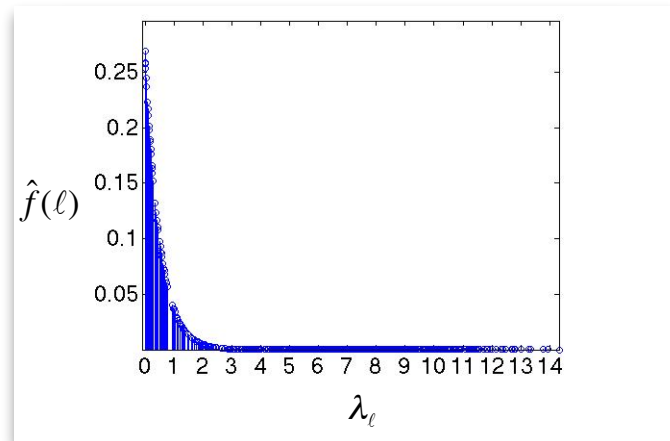
for any admissible kernel



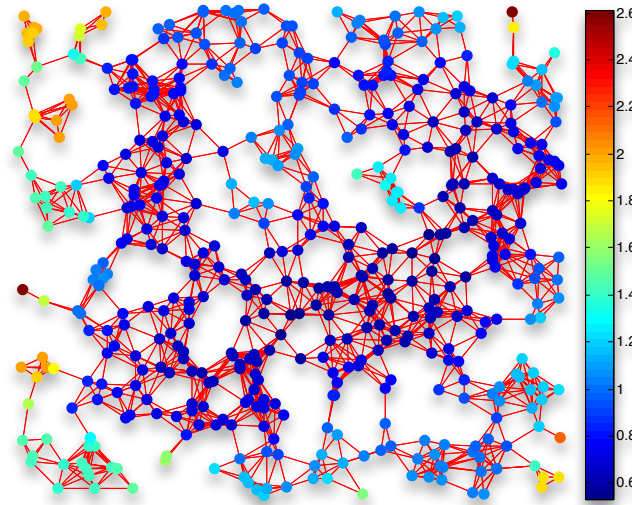
GSP Distance and Centrality

$$d_g(i, j) = \sum_{\ell} (g(\lambda_{\ell}))^2 (u_{\ell}[i] - u_{\ell}[j])^2$$

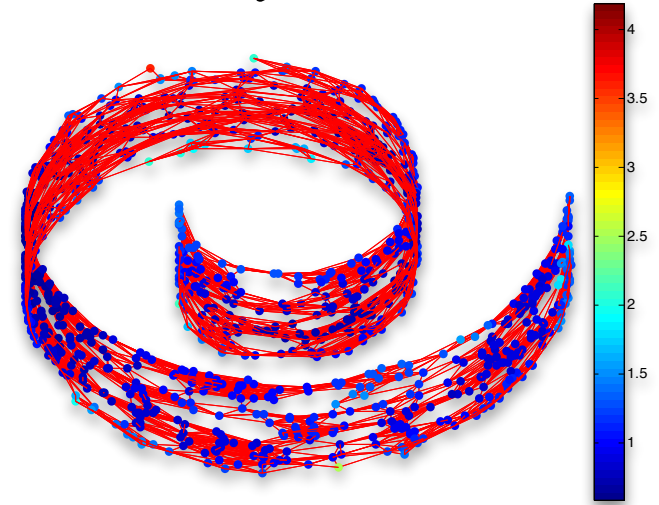
$$\|T_i g\|_2^2 = \sum_{\ell} (g(\lambda_{\ell}))^2 |u_{\ell}[i]|^2$$



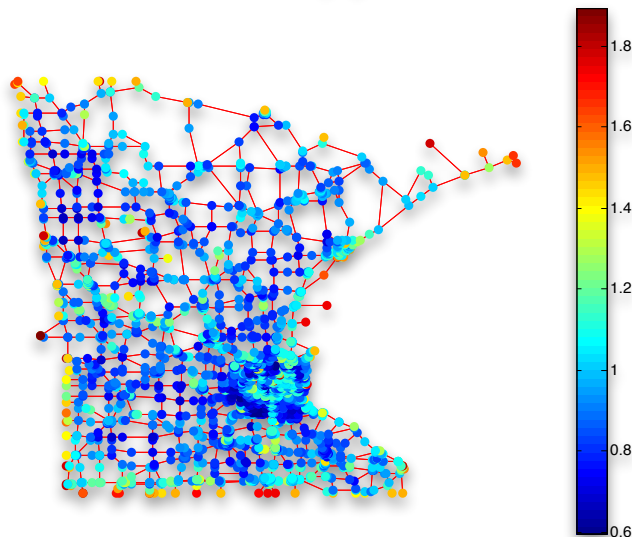
(a)



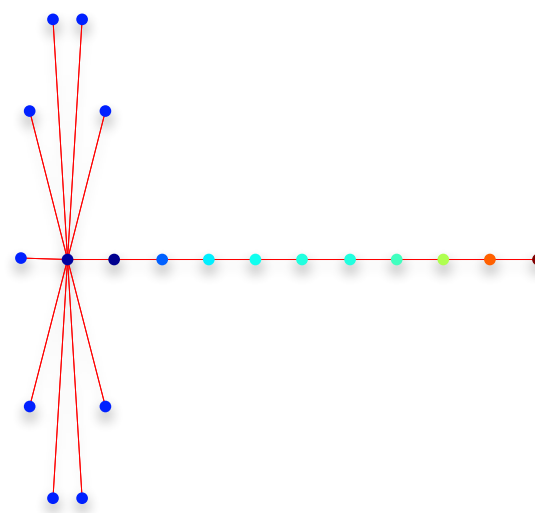
(b)



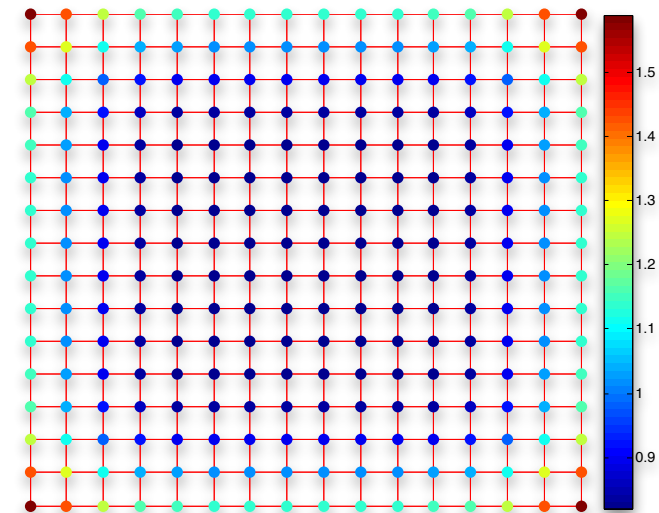
(c)



(d)



(e)

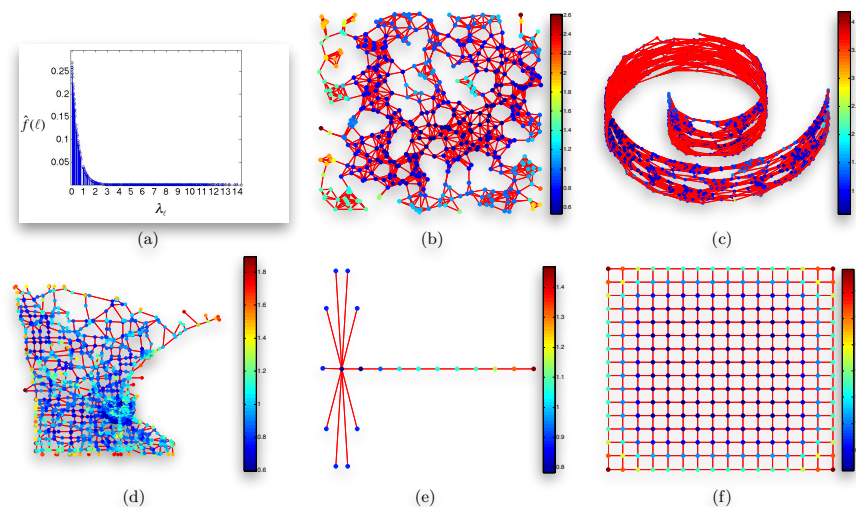


(f)

Structural Node Embeddings with Wavelets

Main idea:

- use graph wavelet as a probe to sense neighbourhood of a node
- construct appropriate feature vector



$$\Psi_s = \psi(sL)$$

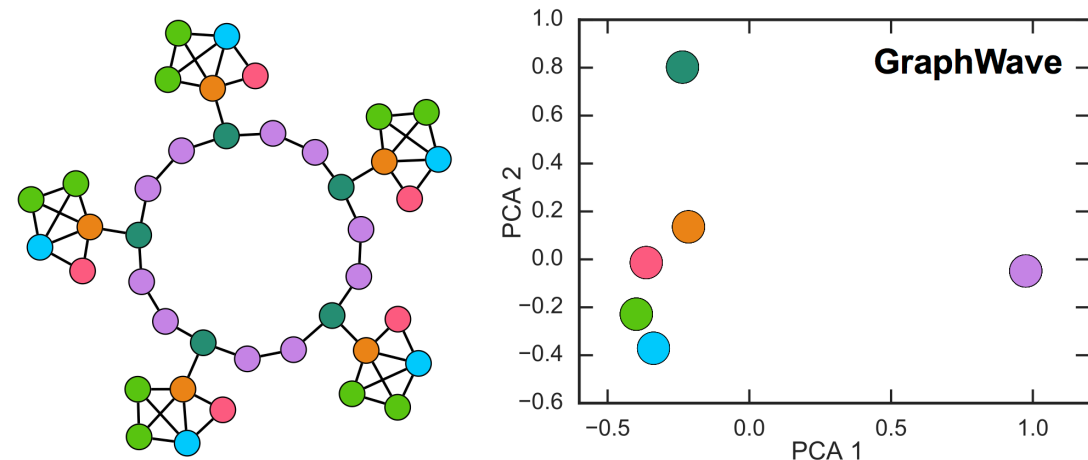
Treat like a probe distribution

Empirical characteristic function

$$\phi_k(t) = \frac{1}{N} \sum_{j=1}^N e^{it\psi_{s,k}(j)}$$

Feature vector = samples of characteristic function

$$z_k = \left[\phi_k(t) \quad \{t = t_1, \dots, t_d\} \right]$$



SP on Graphs Cheat Sheet

$\mathbf{x} \in \mathbb{R}^n$ a (scalar valued) signal

$$\mathbf{L} \in \mathbb{R}^{n \times n}$$

$$\mathbf{L} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^\top$$

Graph Fourier  Frequencies 

Laplacian



$$-\Delta$$



$$-\Delta = \mathcal{F}^T \omega^2 \mathcal{F}$$

Filter and Filtering

$$g(\mathbf{L}) = \mathbf{U} g(\mathbf{\Lambda}) \mathbf{U}^\top$$

$$g(\mathbf{L}) \mathbf{x}$$



$$g \star \mathbf{x}$$

$$\widehat{g(\mathbf{L}) \mathbf{x}}(k) = g(\lambda_k) \hat{\mathbf{x}}(k)$$



$$\widehat{g \star \mathbf{x}}(\omega) = \hat{g}(\omega) \hat{\mathbf{x}}(\omega)$$

A simple interpolation example

Basic interpolation problem (missing features)

Vertex set is split into $\nu = \nu_1 \cup \nu_1^c$

Observe samples $\mathbf{f}_{\nu_1^c}$ of underlying feature map

 Interpolate to recover missing features

Simple model: $\mathbf{f}_{\text{interp}} = \sum_{j \in \mathcal{V}_r} \alpha[j] \varphi_j = \Phi_{\nu_1} \alpha \quad \bar{\mathbf{L}} := \mathbf{L} + \epsilon \mathbf{I}$

$$\varphi_j = T_j g = g(\bar{\mathbf{L}}) \delta_j = \mathbf{U}[g(\boldsymbol{\Lambda})] \mathbf{U}^* \delta_j = \sum_{\ell=0}^{|\mathcal{V}|-1} \frac{1}{\lambda_\ell + \epsilon} u_\ell^*(j) \mathbf{u}_\ell$$

Regularized Green's functions $\bar{\mathbf{L}} \varphi_j = \delta_j$

$$\mathbf{f}_{\nu_1} = \Phi_{\nu_1, \nu_1} \alpha_* \quad \alpha_* = \left[\bar{\mathbf{L}}_{\nu_1, \nu_1} - \bar{\mathbf{L}}_{\nu_1, \nu_1^c} (\bar{\mathbf{L}}_{\nu_1^c, \nu_1^c})^{-1} \bar{\mathbf{L}}_{\nu_1^c, \nu_1} \right] \mathbf{f}_{\nu_1}$$

Simplified version of Pesenson's variational spline interpolation

A simple interpolation example

Graph filtering interpretation of the algorithm:

$$\alpha_* = \left[\bar{\mathbf{L}}_{\mathcal{V}_1, \mathcal{V}_1} - \bar{\mathbf{L}}_{\mathcal{V}_1, \mathcal{V}_1^c} (\bar{\mathbf{L}}_{\mathcal{V}_1^c, \mathcal{V}_1^c})^{-1} \bar{\mathbf{L}}_{\mathcal{V}_1^c, \mathcal{V}_1} \right] \mathbf{f}_{\mathcal{V}_1}$$

Kron-reduced regularised Laplacian

Upsample coefficients as graph signal (set of weighted kroneckers):

$$\mathbf{f}_{\text{upsamp}} = \sum_{j \in \mathcal{V}_1} \alpha_*[j] \delta_j$$

Compute interpolation by filtering:

$$\mathbf{f}_{\text{interp}} = g(\bar{\mathbf{L}}) \mathbf{f}_{\text{upsamp}}$$

You can compress smooth feature fields and reconstruct them efficiently

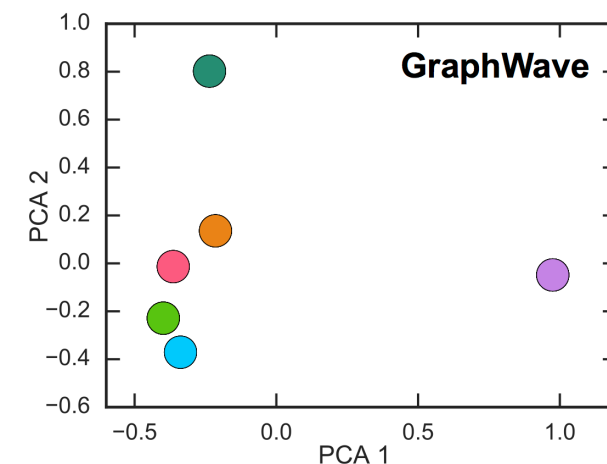
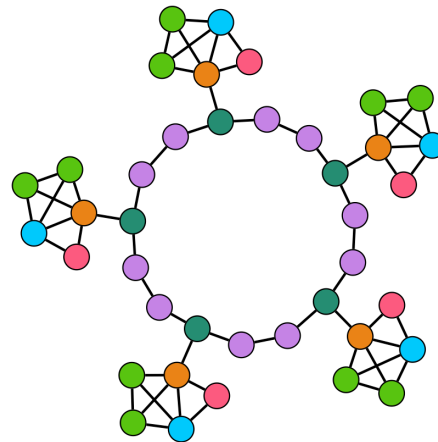
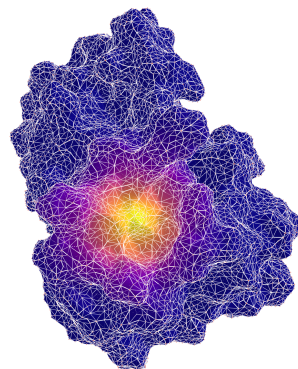
Previously on Graph Rep Learning

Eigen decomposition of Laplacian reveals structure of graph

Ingredient: Smoothness of feature vectors (smooth partition signals)

Understand, analyse, process graph signals. Graph filters.

Crafting specific features, de-noising, interpolating missing features ...



Summary: **LOVE** is all you need

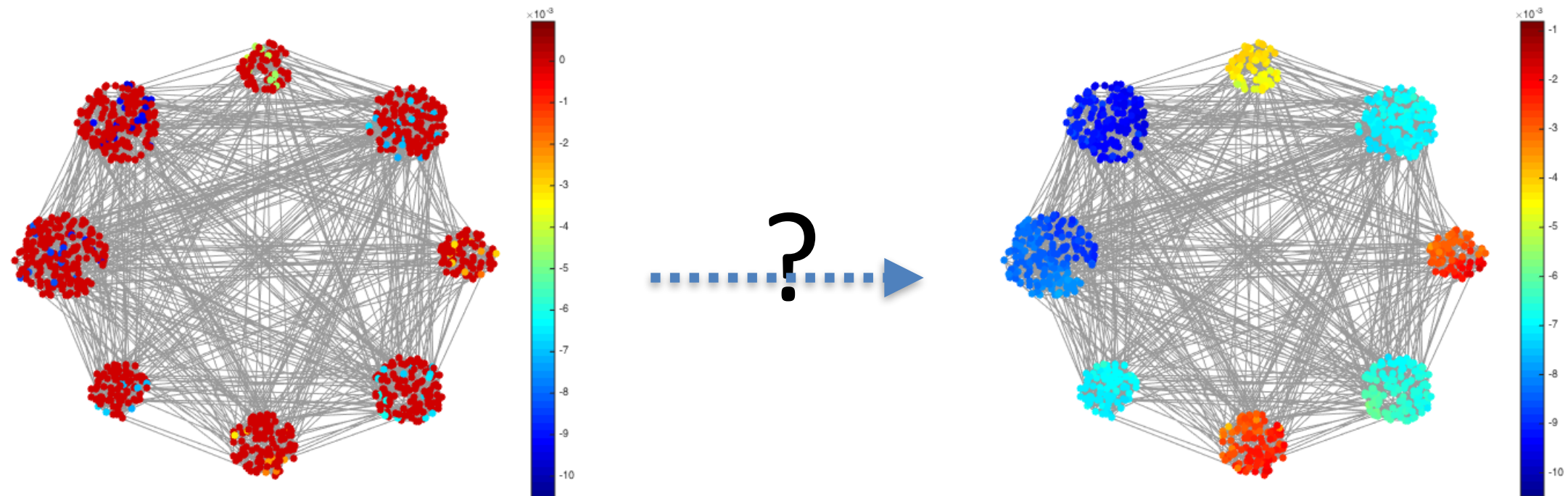
Laplacian **O**rthogonal eigen**V**ectors

Back to Spectral Clustering

Compressive Spectral Clustering, Tremblay, Puy, Gribonval, VDG, ICML'16

Random sampling of band limited signals on graphs, ACHA 2015

Given partially observed information at the nodes of a graph



Can we robustly and efficiently infer missing information ?

What signal model ?

How many observations ?

Influence of the structure of the graph ?

Notations

k -bandlimited signals $\mathbf{x} \in \mathbb{R}^n$

Fourier coefficients $\hat{\mathbf{x}} = \mathbf{U}^\top \mathbf{x}$

$$\mathbf{x} = \mathbf{U}_k \hat{\mathbf{x}}^k \quad \hat{\mathbf{x}}^k \in \mathbb{R}^k$$

$$\mathbf{U}_k := (\mathbf{u}_1, \dots, \mathbf{u}_k) \in \mathbb{R}^{n \times k} \quad \text{first } k \text{ eigenvectors only}$$

k -bandlimited signals are smooth over the graph

$$\begin{aligned} \mathbf{x}^\top \mathbf{L} \mathbf{x} &= \sum_{i \sim j} w_{ij} (\mathbf{x}[i] - \mathbf{x}[j])^2 \\ &= \sum_k \lambda_k |\hat{\mathbf{x}}^k|^2 \end{aligned}$$

Sampling Model

$$\mathbf{p} \in \mathbb{R}^n \quad \mathbf{p}_i > 0 \quad \|\mathbf{p}\|_1 = \sum_{i=1}^n \mathbf{p}_i = 1$$

$$\mathbf{P} := \text{diag}(\mathbf{p}) \in \mathbb{R}^{n \times n}$$

Draw independently m samples (random sampling)

$$\mathbb{P}(\omega_j = i) = \mathbf{p}_i, \quad \forall j \in \{1, \dots, m\} \text{ and } \forall i \in \{1, \dots, n\}$$

$$\mathbf{y}_j := \mathbf{x}_{\omega_j}, \quad \forall j \in \{1, \dots, m\}$$

$$\mathbf{y} = \mathbf{M}\mathbf{x}$$

Stable Embedding

Sampling density that guarantees information is preserved

Variable Density Sampling

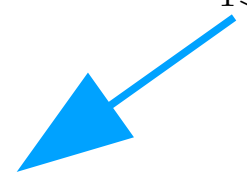
$$\mathbf{p}_i^* := \frac{\|\mathbf{U}_k^\top \boldsymbol{\delta}_i\|_2^2}{k}, \quad i = 1, \dots, n$$

Corollary 1. *Let $\mathbf{M}$ be a random subsampling matrix constructed with the sampling distribution $\mathbf{p}^*$. For any $\delta, \epsilon \in (0, 1)$, with probability at least $1 - \epsilon$,*

$$(1 - \delta) \|\mathbf{x}\|_2^2 \leq \frac{1}{m} \left\| \mathbf{M} \mathbf{P}^{-1/2} \mathbf{x} \right\|_2^2 \leq (1 + \delta) \|\mathbf{x}\|_2^2$$

for all $\mathbf{x} \in \text{span}(\mathbf{U}_k)$ provided that

$$m \geq \frac{3}{\delta^2} k \log \left(\frac{2k}{\epsilon} \right).$$

$$\nu_k = \sqrt{N} \max_{1 \leq i \leq N} \{\|\mathbf{U}_k^\top \boldsymbol{\delta}_i\|_2\} \quad \sqrt{k} \leq \nu_k \leq \sqrt{N}$$


Rem: k can be estimated by eigencount = filtering random signals !

(Di Napoli et al, Efficient estimation of eigenvalue counts in an interval)

Recovery Procedures (Inference)

$$\mathbf{y} = \mathbf{M}\mathbf{x} + \mathbf{n} \quad \mathbf{y} \in \mathbb{R}^m$$

$$\mathbf{x} \in \text{span}(\mathbf{U}_k) \quad \text{stable embedding}$$

Efficient Decoder:

$$\min_{\mathbf{z} \in \mathbb{R}^n} \left\| \mathbf{P}_{\Omega}^{-1/2} (\mathbf{M}\mathbf{z} - \mathbf{y}) \right\|_2^2 + \gamma \mathbf{z}^{\top} g(\mathbf{L}) \mathbf{z}$$

one can construct an
optimal sampling distribution
from the graph only

soft constrain on frequencies
efficient implementation

Analysis of Efficient Decoder

Noiseless case:

$$\|\mathbf{x}^* - \mathbf{x}\|_2 \leq \frac{1}{\sqrt{m(1-\delta)}} \left(M_{\max} \sqrt{\frac{g(\boldsymbol{\lambda}_k)}{g(\boldsymbol{\lambda}_{k+1})}} + \sqrt{\gamma g(\boldsymbol{\lambda}_k)} \right) \|\mathbf{x}\|_2 + \sqrt{\frac{g(\boldsymbol{\lambda}_k)}{g(\boldsymbol{\lambda}_{k+1})}} \|\mathbf{x}\|_2$$

$g(\boldsymbol{\lambda}_k) = 0$ + non-decreasing implies perfect reconstruction

Otherwise:

choose γ as close as possible to 0 and seek to minimise the ratio $g(\boldsymbol{\lambda}_k)/g(\boldsymbol{\lambda}_{k+1})$

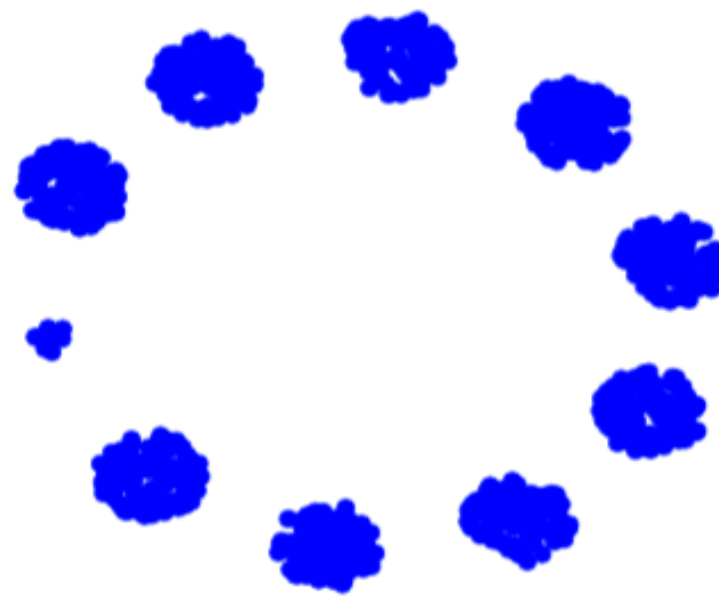
Choose filter to increase spectral gap ?

Clusters are of course good

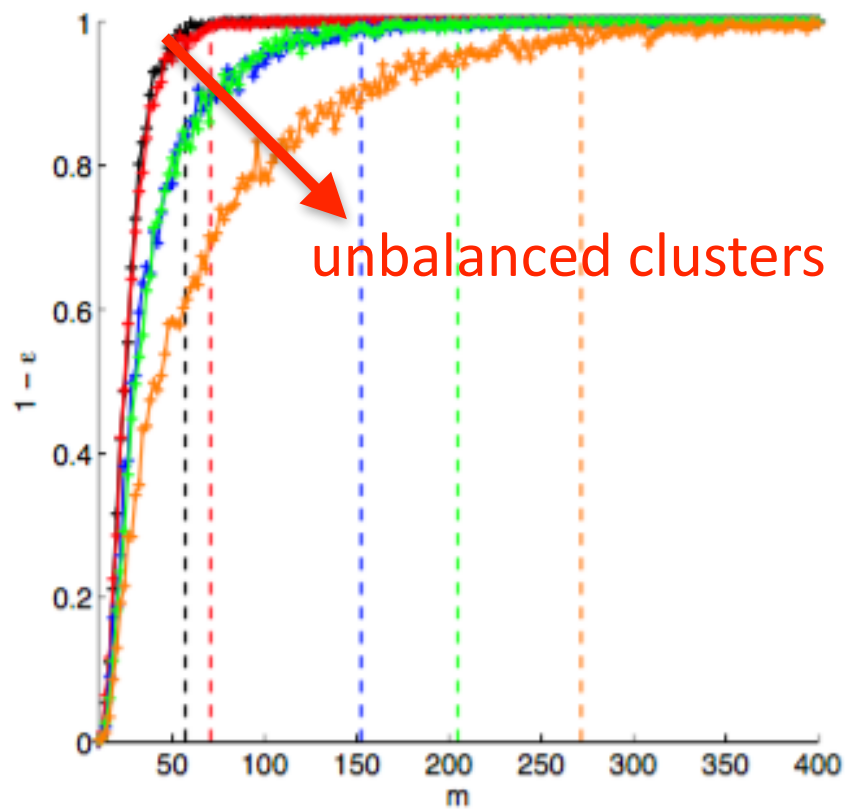
Noise: $\|\mathbf{P}_{\Omega}^{-1/2} \mathbf{n}\|_2 / \|\mathbf{x}\|_2$

Experiments

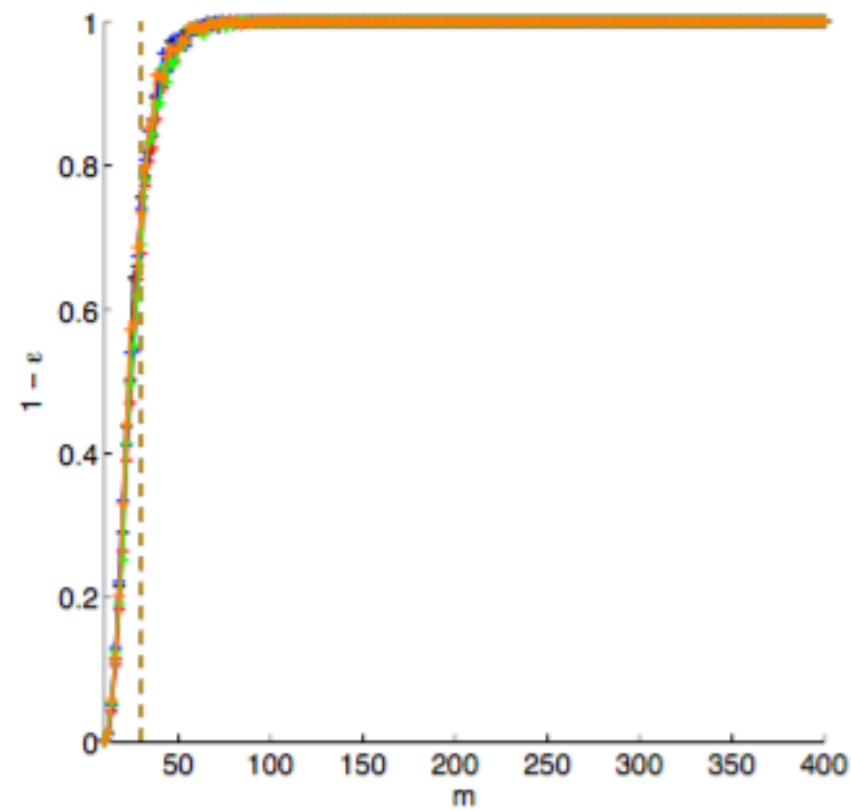
Community graph



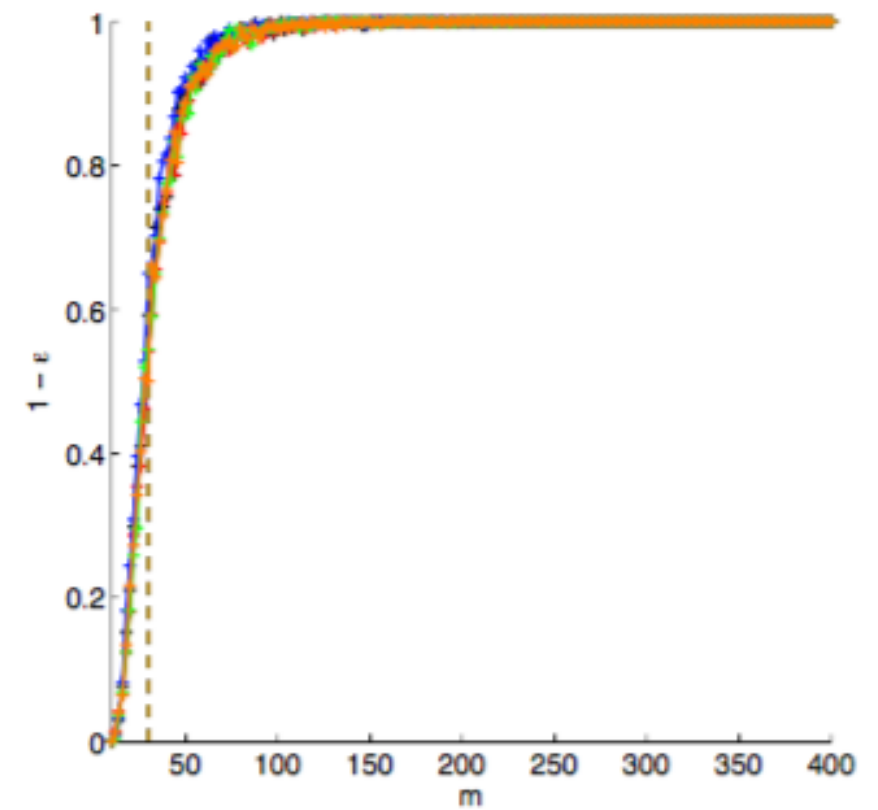
Uniform distribution π



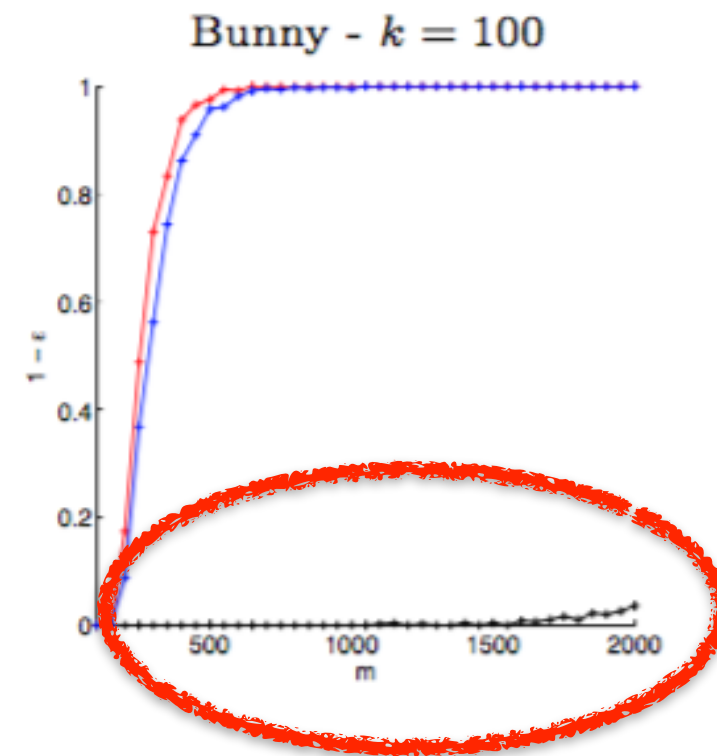
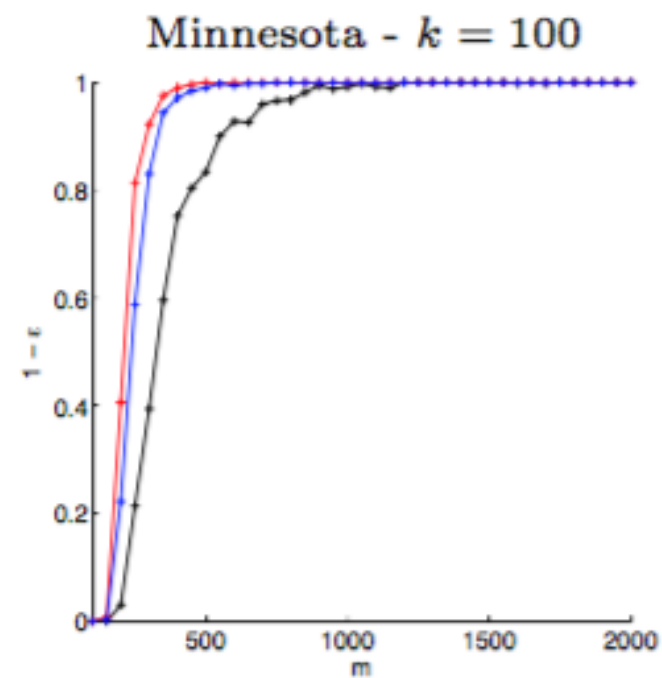
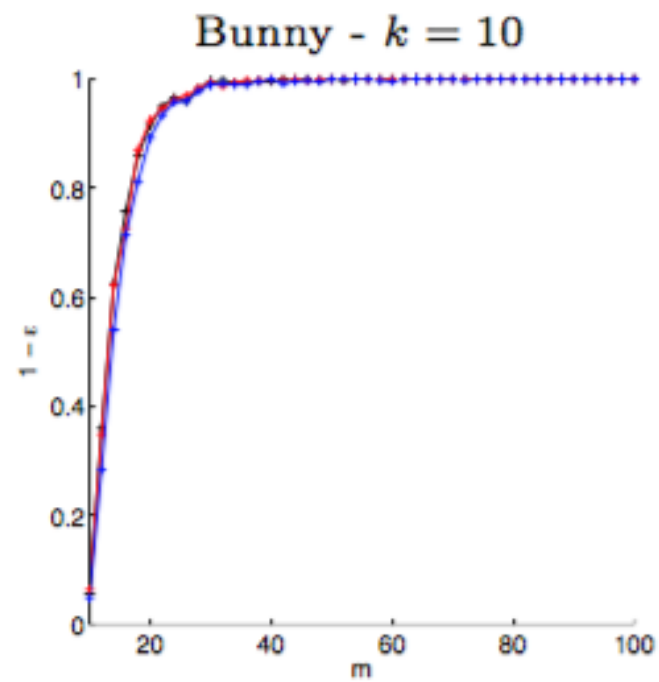
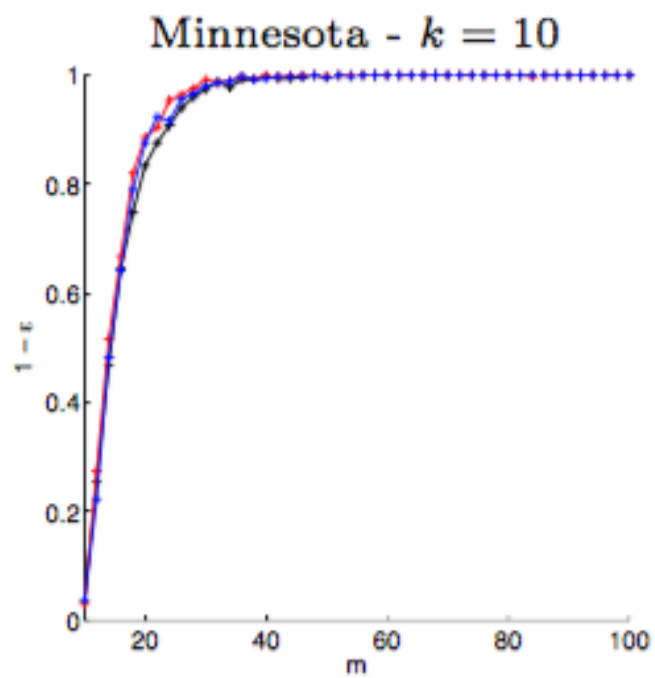
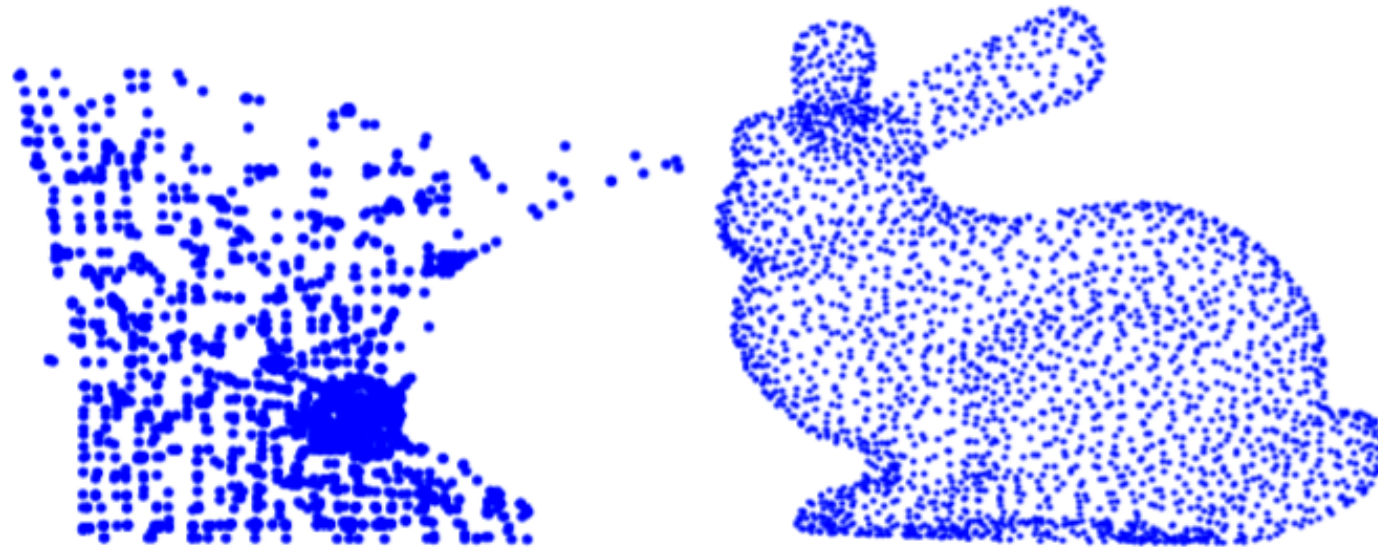
Optimal distribution p^*



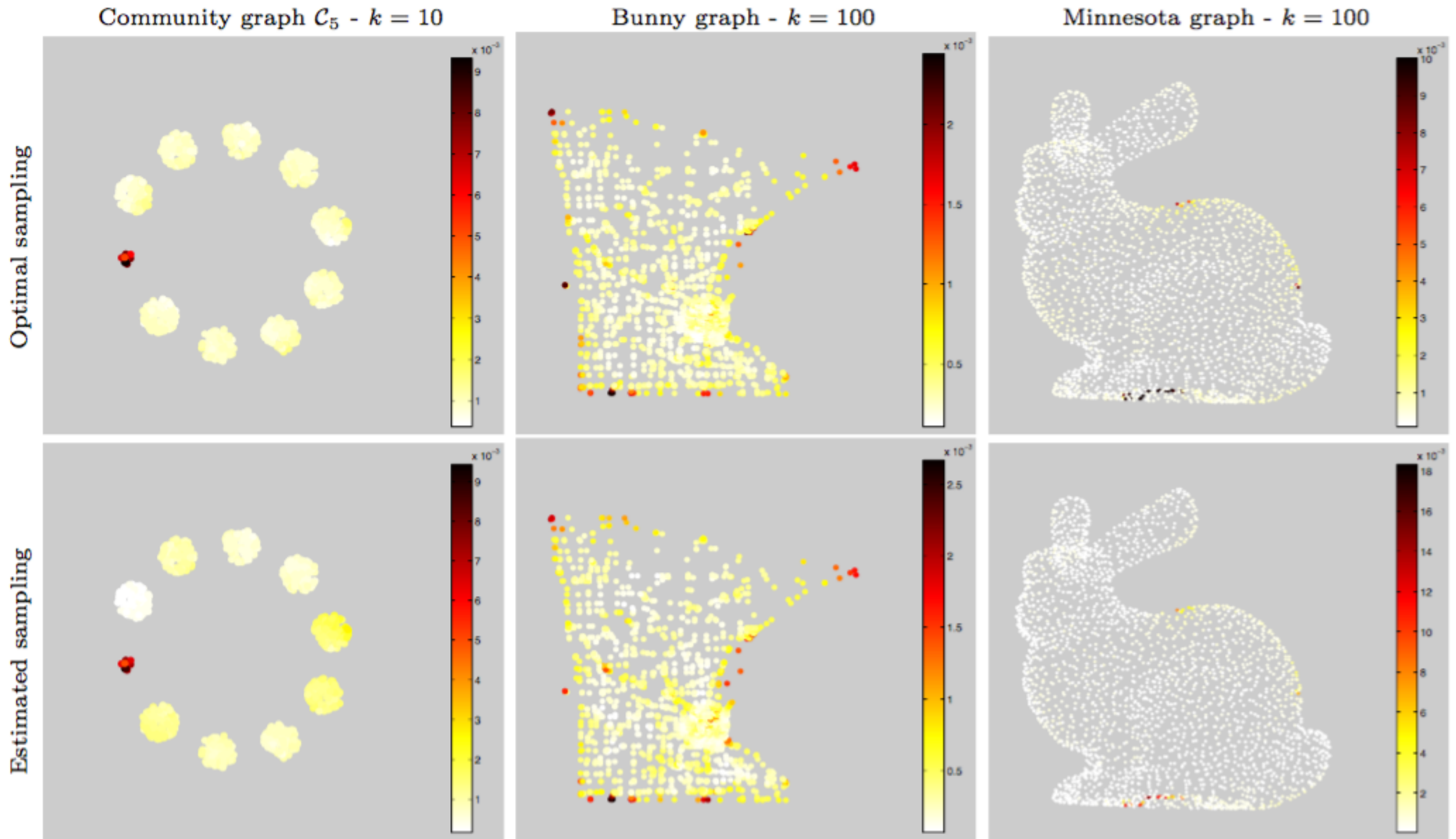
Estimated distribution $\tilde{p}$



Experiments



Experiments



Experiments

Original



Reconstructed (sampling with $\tilde{p}$)



Reconstructed (sampling with π)



Compressive Spectral Clustering

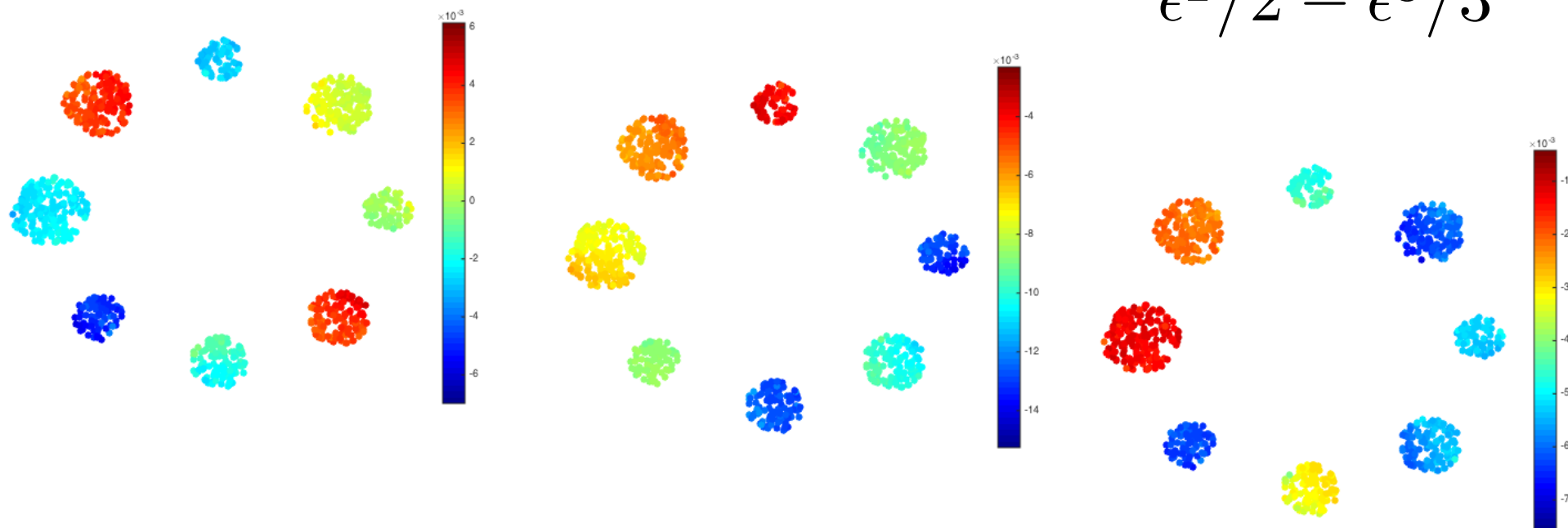
Clustering equivalent to recovery of cluster assignment functions

Well-defined clusters $\rightarrow$ band-limited assignment functions!

Generate features by filtering random signals

by Johnson-Lindenstrauss

$$\eta = \frac{4 + 2\beta}{\epsilon^2/2 - \epsilon^3/3} \log n$$



Compressive Spectral Clustering

Clustering equivalent to recovery of cluster assignment functions

Well-defined clusters $\rightarrow$ band-limited assignment functions!

Generate features by filtering random signals

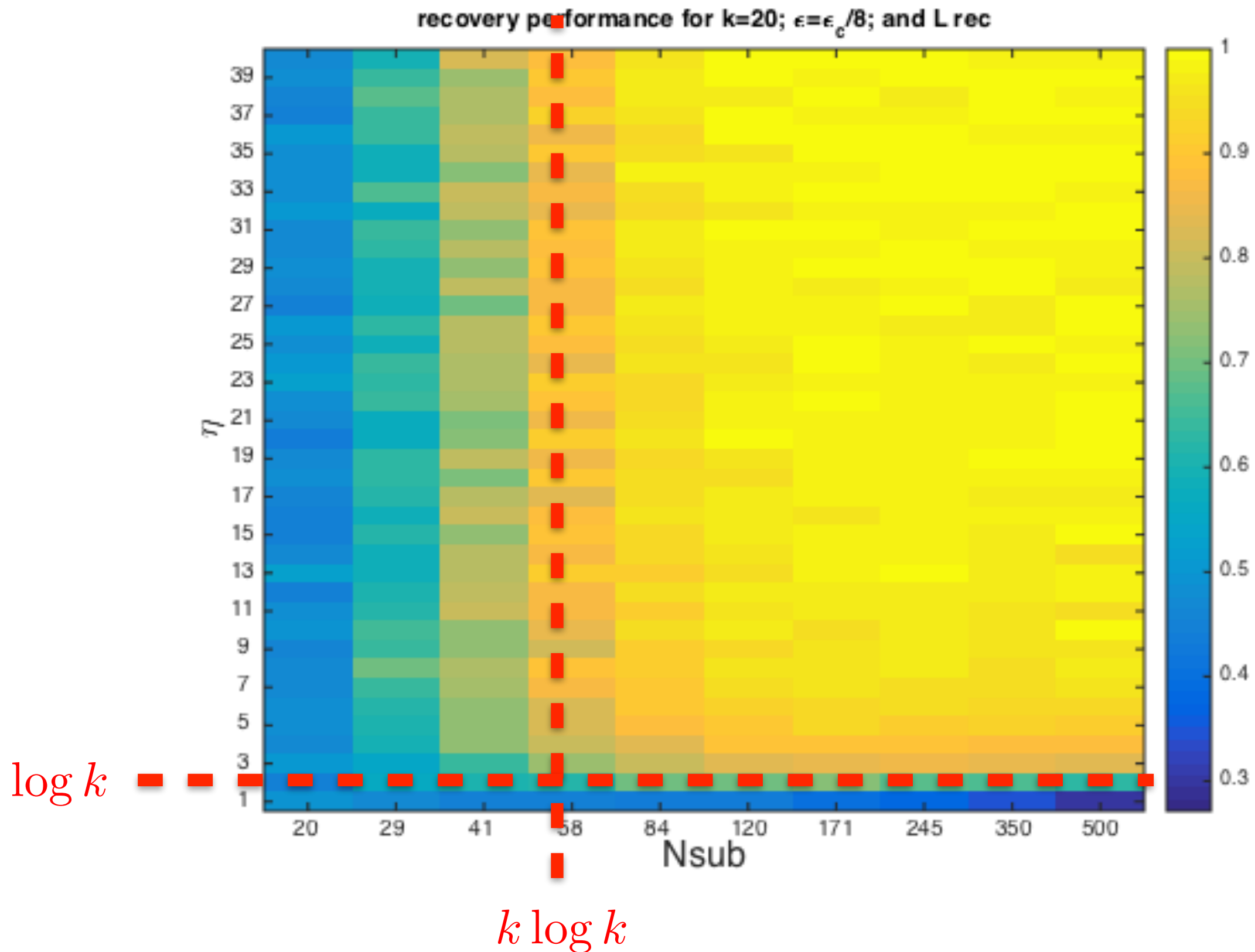
by Johnson-Lindenstrauss
$$\eta = \frac{4 + 2\beta}{\epsilon^2/2 - \epsilon^3/3} \log n$$

Each feature map is smooth, therefore keep

$$m \geq \frac{6}{\delta^2} \nu_k^2 \log \left(\frac{k}{\epsilon'} \right)$$

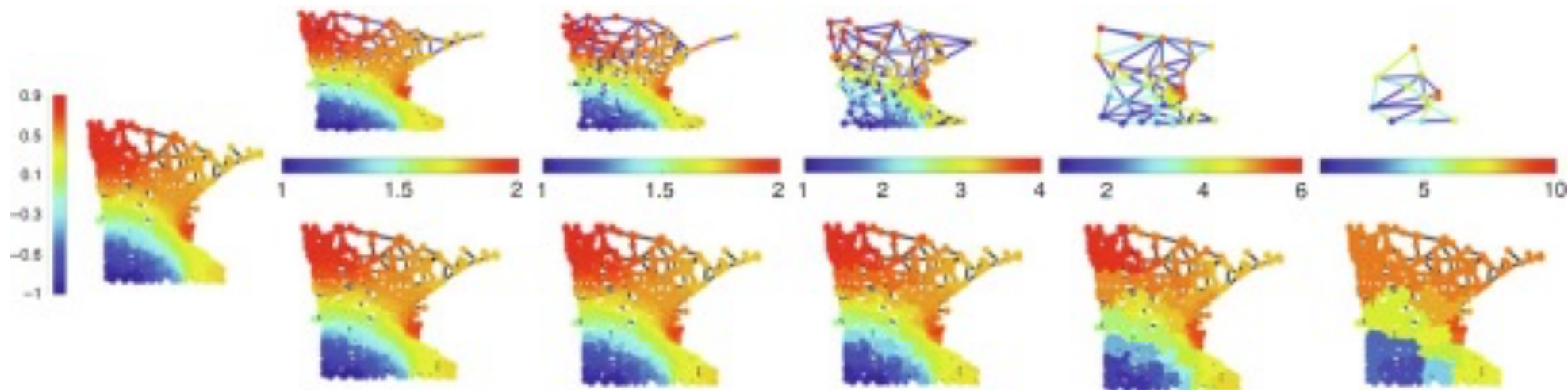
Use k-means on compressed data and feed into Efficient Decoder

Compressive Spectral Clustering

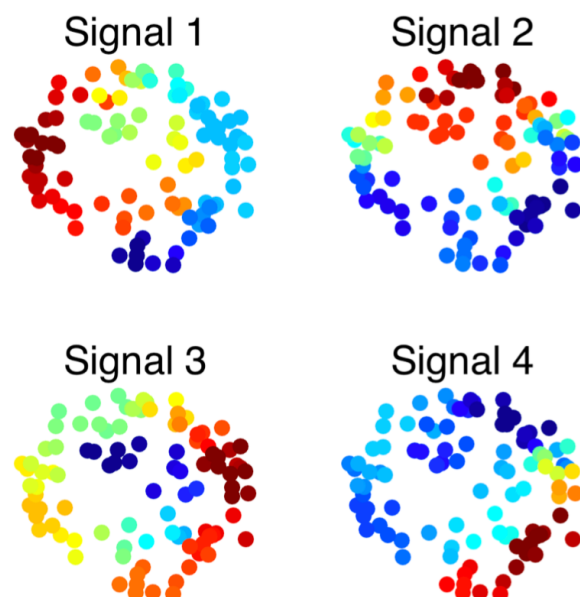


Other developments

Filterbanks, wavelets and other transforms on graphs



Stationary graph signal processing
(statistical “shift” invariance)



but also sampling, compression ...