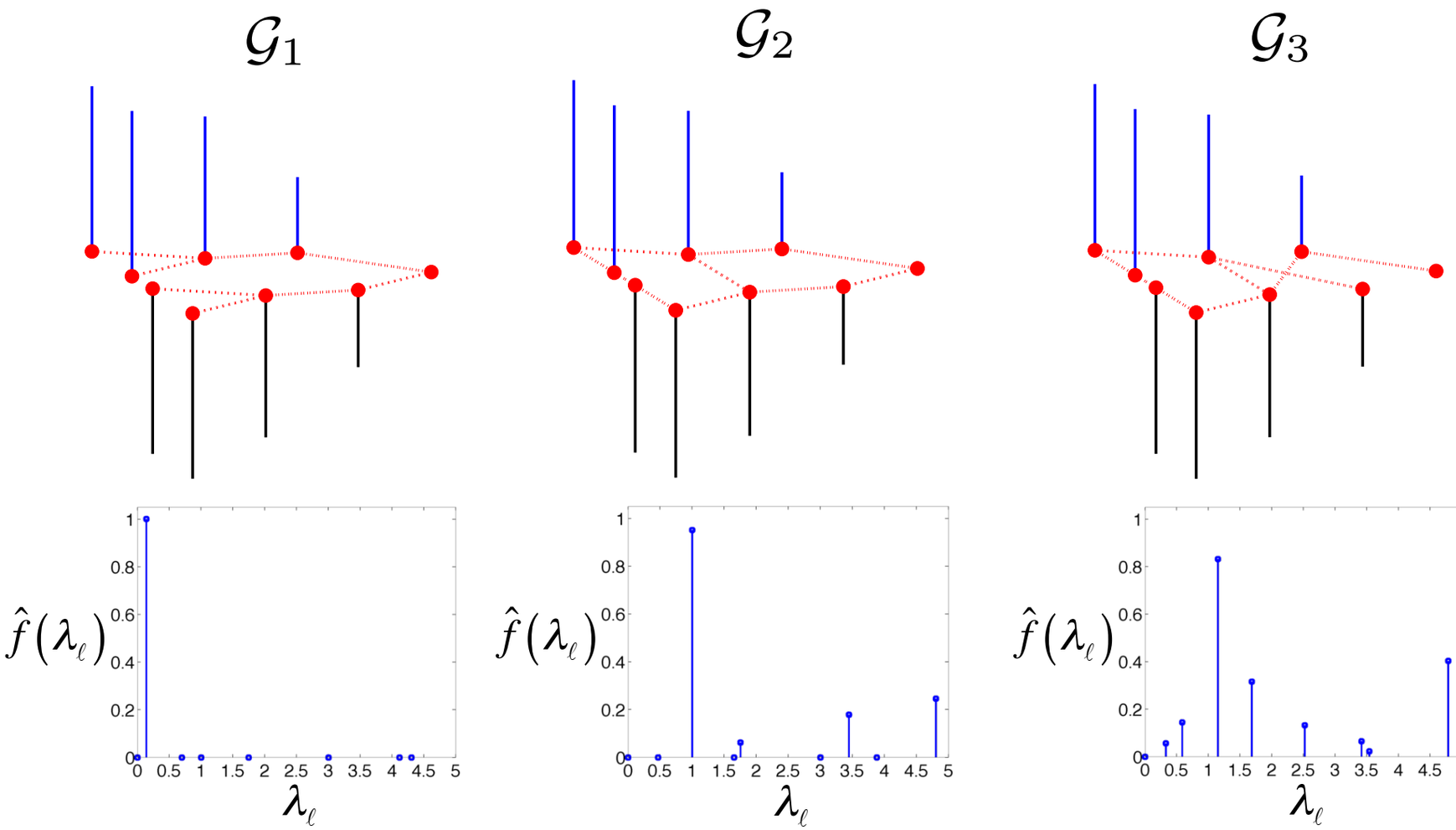
Processing Data over Graphs

The Graph Fourier Transform

Using eigenvectors of $\mathbf{L}$, we define a Graph Fourier Transform (GFT)



$$f^T \mathbf{L}_1 f = 0.14 \qquad f^T \mathbf{L}_2 f = 1.31 \qquad f^T \mathbf{L}_3 f = 1.81$$

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 (Borel) functional calculus

Borel 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

From polynomial to continuous functions by Stone-Weierstrass

Then Riesz-Markov (non-trivial !)

Example: Diffusion on Graphs

Consider the following « heat » diffusion model

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

$$\hat{f}(\ell, t) = e^{-t\lambda_\ell}\hat{f}_0(\ell) \qquad f = e^{-t\mathcal{L}} f_0 \qquad \text{by functional calculus}$$

Explicitly:

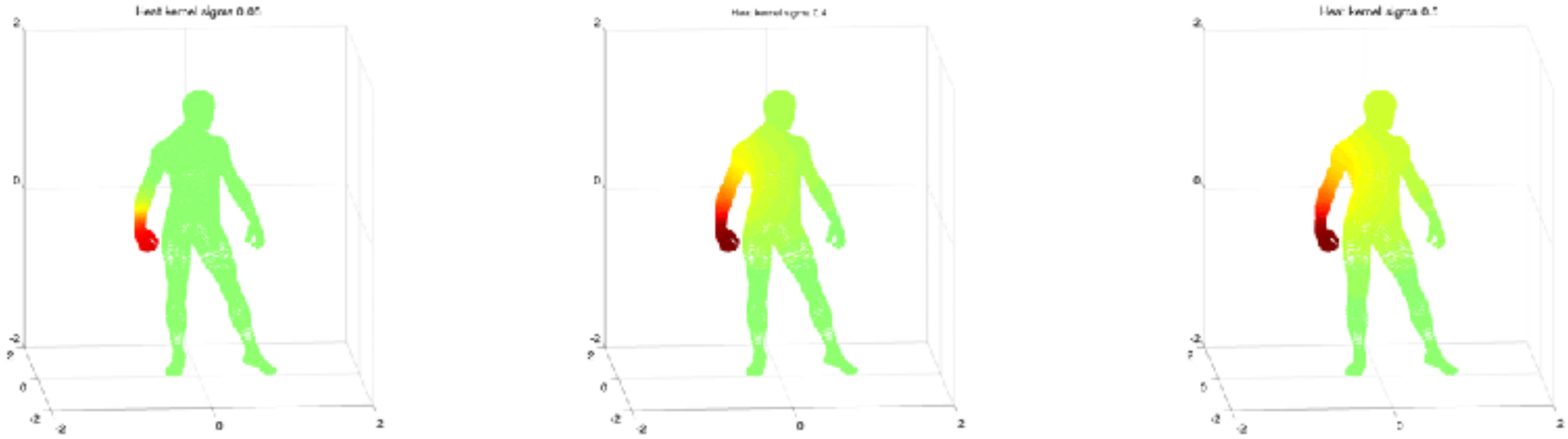
$$f(i) = \sum_{j \in V} \sum_\ell e^{-t\lambda_\ell} u_\ell(i) u_\ell(j) f_0(j)$$

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

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

Example: Diffusion on Graphs

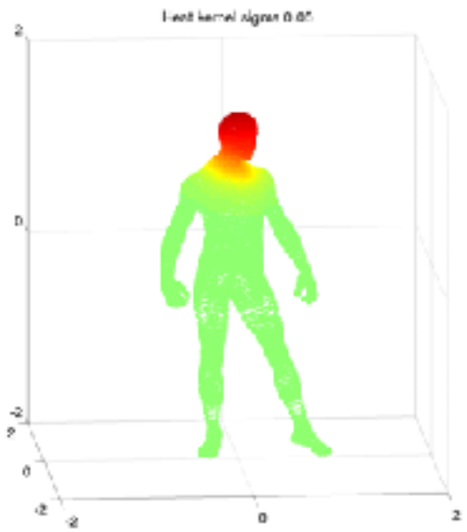
examples of heat kernel on graph



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

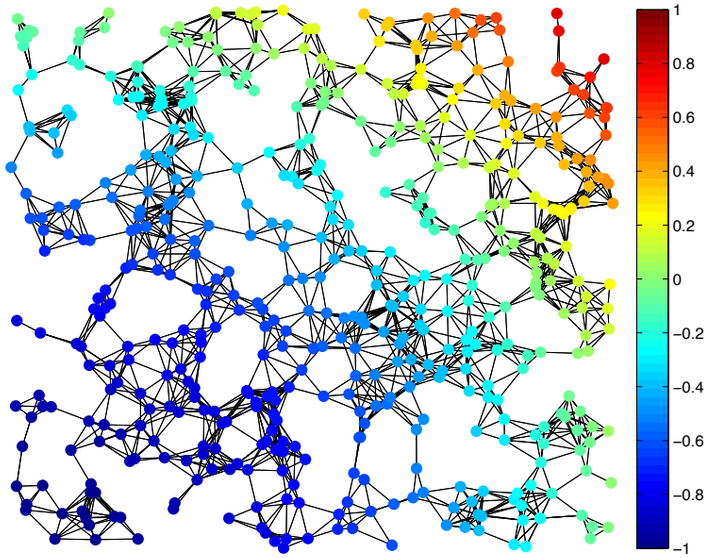
$$f(i) = \sum_\ell e^{-t\lambda_\ell} \hat{f}_0(\ell) u_\ell(i)$$

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



Simple De-Noising Example

Suppose a smooth signal f on a graph



Original

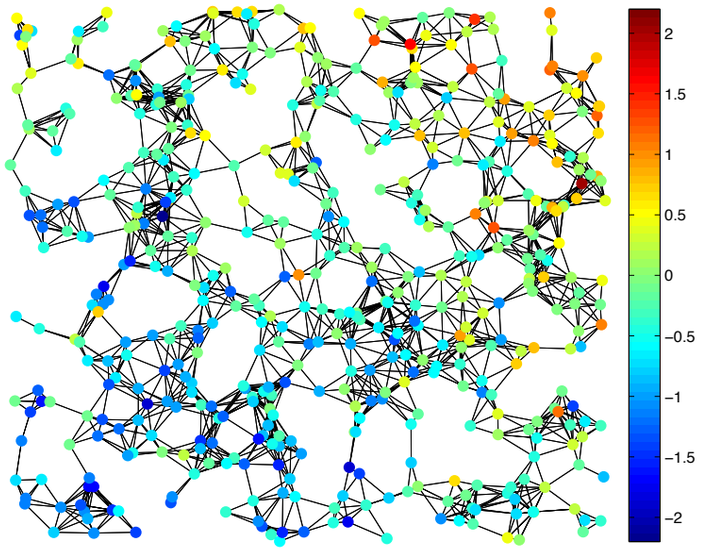
$$\|\nabla f\|_2^2 \leq M \Leftrightarrow f^t \mathbf{L} f \leq M$$

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

But you observe only a noisy version y

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

Noisy



Simple De-Noising Example

De-Noising by Regularization

$$\underset{f}{\operatorname{argmin}} \|f - y\|_2^2 \text{ s.t. } f^t \mathbf{L} f \leq M$$

$$\underset{f}{\operatorname{argmin}} \frac{\tau}{2}\|f - y\|_2^2 + f^{\mathrm{T}} \mathcal{L}^r f \quad \Rightarrow \quad \mathcal{L}^r f_* + \frac{\tau}{2}(f_* - y) = 0$$

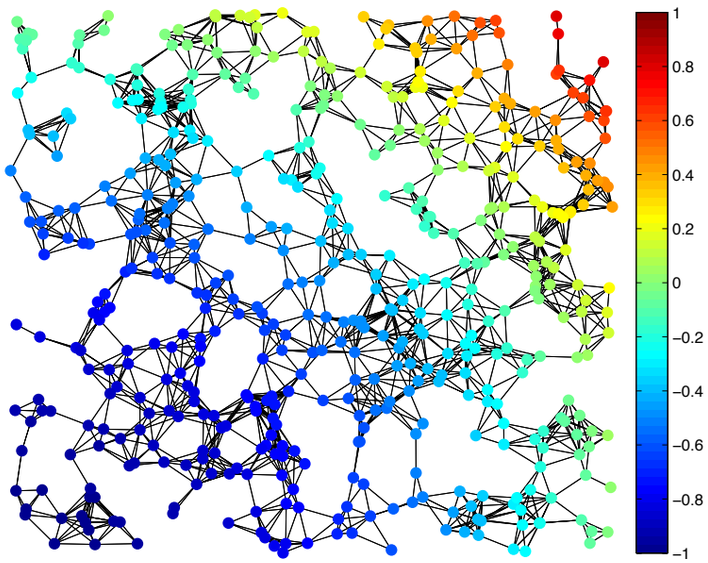
Graph Fourier

$$\Rightarrow \quad \widehat{\mathcal{L}^r f_*}(\ell) + \tfrac{\tau}{2}\left(\widehat{f}_*(\ell) - \hat{y}(\ell)\right) = 0, \quad \forall \ell \in \{0, 1, \ldots, N-1\}$$

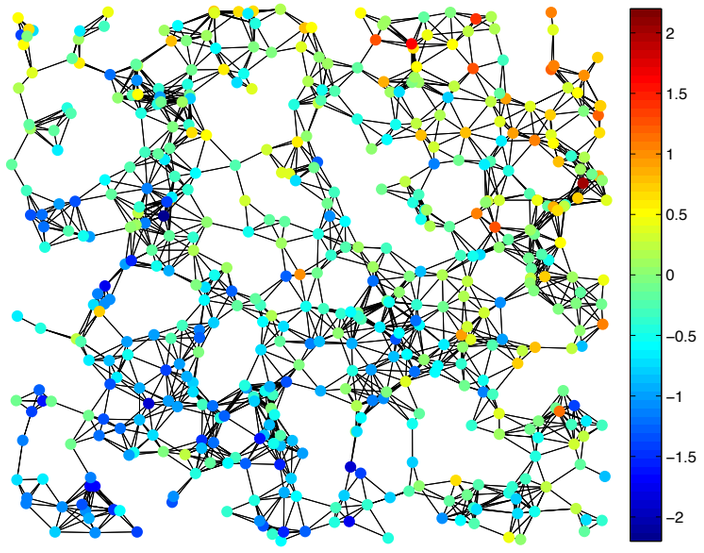
$$\Rightarrow \quad \widehat{f}_*(\ell) = \frac{\tau}{\tau + 2\lambda_\ell^r} \hat{y}(\ell) \qquad \text{“Low pass” filtering !}$$

Simple De-Noising Example

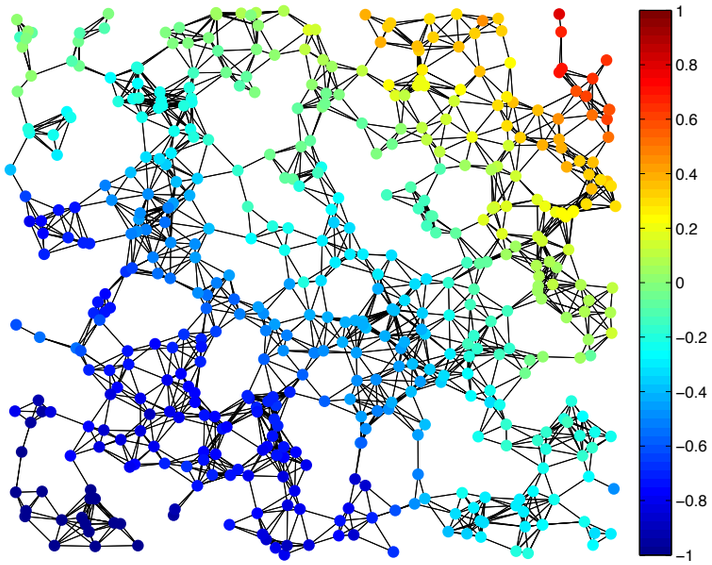
$$\text{argmin}_f \left\{ ||f - y||_2^2 + \gamma f^T \mathcal{L} f \right\}$$



Original



Noisy



Denoised

$$\underset{f}{\text{argmin}} \frac{\tau}{2} \|f - y\|_2^2 + f^{\mathrm{T}} \mathcal{L}^r f \quad \Rightarrow \quad \mathcal{L}^r f_* + \frac{\tau}{2}(f_* - y) = 0$$

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

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

Filtering Data on Graphs

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 often called "convolution over graphs"

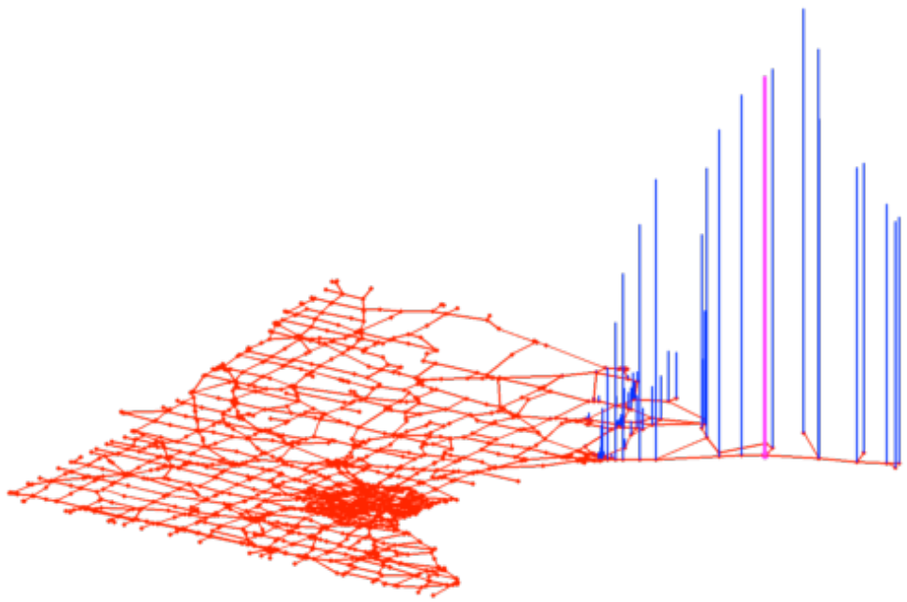
Graph Filters as Features

Filtering = Correlation with a localised pattern

$$\big(g(\mathbf{L})f\big)[i] = \sum_\ell g(\lambda_\ell)\hat{f}(\lambda_\ell)u_\ell[i]$$

$$= \sum_j f[j] \sum_\ell g(\lambda_\ell)u_\ell[i]u_\ell[j]$$

$$= \langle T_i g, f\rangle \longrightarrow \big(T_i g\big) = g(\mathbf{L})\delta_i$$



Polynomial Localization

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

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

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:

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

Are these features localized ?

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

Construct an order K polynomial approximation:

$$\phi'_n[m] = \langle \delta_m, P_K(\mathbf{L})\delta_n \rangle$$

Exactly localized in a K-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:

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

Application: Learning over Graphs

Let X be an array of data points $\quad x_1, x_2, ..., x_n \in \mathbb{R}^d$

Each point has a desired class label $y_k \in Y$ (suppose binary)

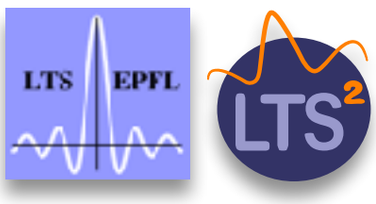
At training you have the labels of a subset S of X $\quad |S| = l < n$

Getting data is easy but labeled data is a scarce resource

GOAL: predict remaining labels

Rationale: minimize empirical risk on your training data such that

- your model is predictive
- your model is simple, does not overfit
- your model is "stable" (depends continuously on your training set)
- ...



Linear Regression Learning

Ex: Linear regression $y_k = \beta \cdot x_k + b$

Empirical Risk: $\|\mathbf{X}^t\beta - \mathbf{y}\|_2^2 \Longrightarrow \beta = (\mathbf{X}\mathbf{X}^t)^{-1}X\mathbf{y}$

if not enough observations, regularize (Tikhonov):

$$\|\mathbf{X}^t\beta - \mathbf{y}\|_2^2 + \alpha\|\beta\|_2^2 \Longrightarrow \beta = (\mathbf{X}\mathbf{X}^t + \alpha\mathbf{I})^{-1}X\mathbf{y}$$

Ridge Regression

Questions:

How can unlabelled data be used ?

More general linear model with a dictionary of features ?

$$\arg\min_{\beta} \|\mathbf{y} - \mathbf{M}\mathbf{\Phi}_X\beta\|_2^2 + \alpha\mathcal{S}(\beta)$$

dictionary depends on data points

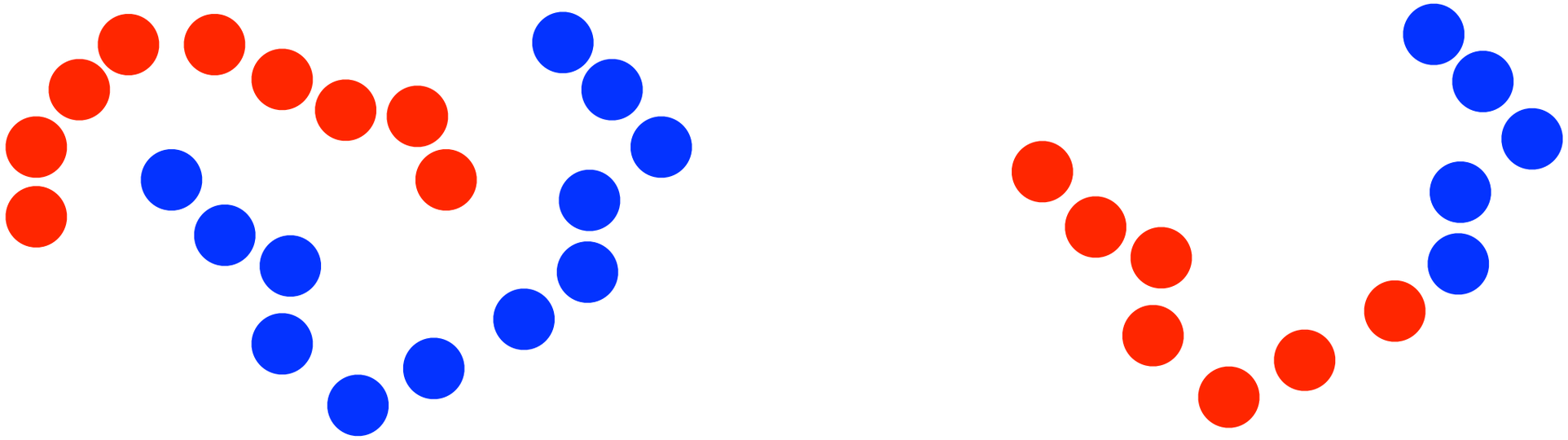
simplifies/stabilizes selected model

Learning on Graphs

How can unlabelled data be used ?

Assumption:

target function is not globally smooth but it is locally smooth over regions of data space that have some geometrical structure



Use graph to model this structure

Learning on with Graphs

Example (Belkin, Niyogi)

Affinity between data points represented by edge weights (affinity matrix W)

measure of smoothness: $\|\nabla f\|_2^2 = \sum_{i,j \in X} \mathbf{W}_{ij}(f(x_i) - f(x_j))^2$

$= f^t \mathbf{L} f$

Revisit ridge regression: $\|\mathbf{X}_S^t \beta - \mathbf{y}\|_2^2 + \alpha \|\beta\|_2^2 + \gamma \beta^t \mathbf{X} L \mathbf{X}^t \beta$

Regression part on labelled data

Solution is smooth in graph "geometry" (all data)

Learning on with Graphs

More general linear model with a dictionary of features ?

$\mathbf{\Phi}_X$ dictionary of features on the complete data set (data dependent)

$\boldsymbol{M}$ restricts to labeled data points (mask)

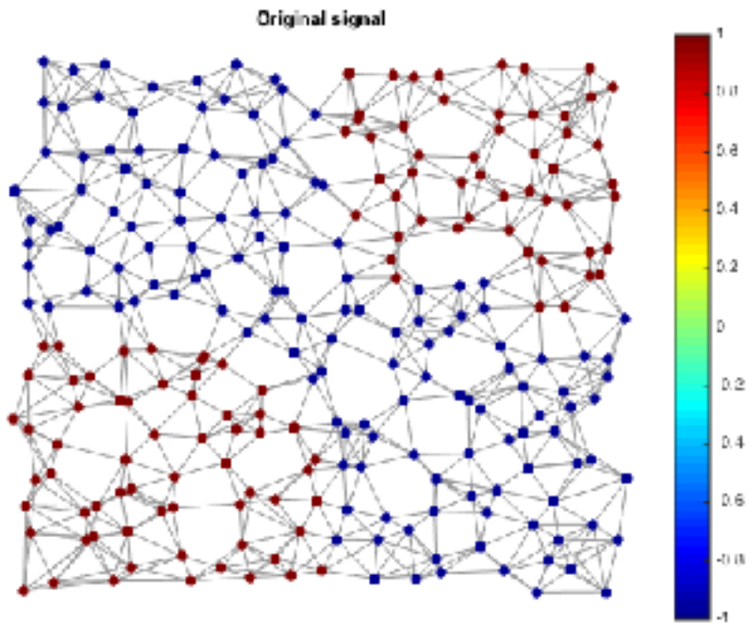
$$\arg\min_{\beta} \|\mathbf{y} - \mathbf{M}\mathbf{\Phi}_X\beta\|_2^2 + \alpha\mathcal{S}(\beta)$$

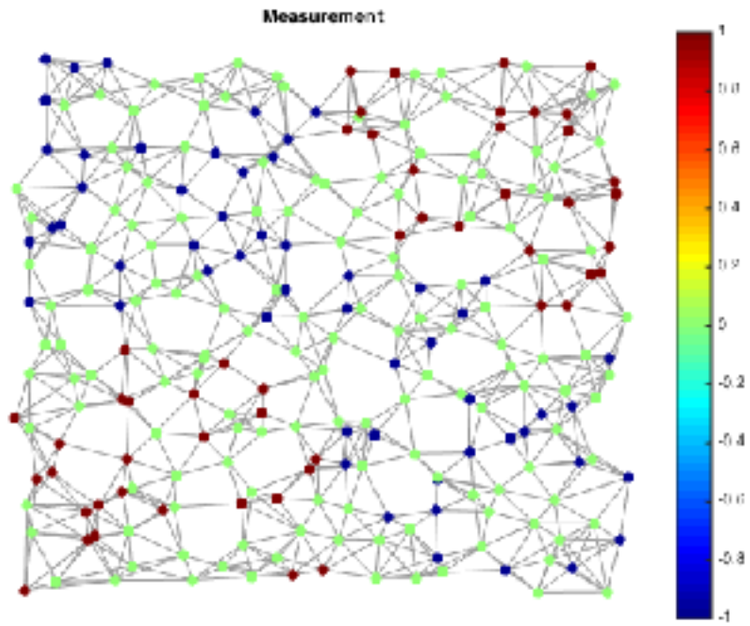
Empirical Risk

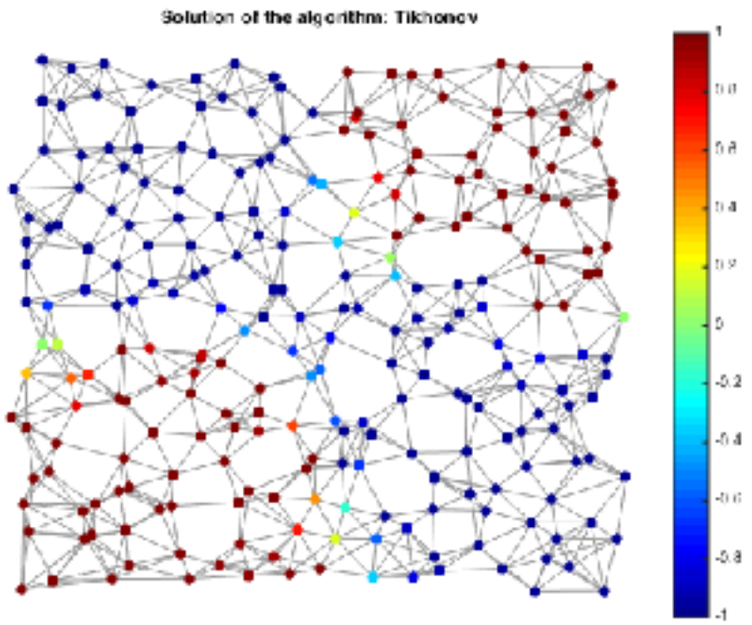
Model Selection penalty, sparsity ?
Smoothness on graph ?

Important Note: our dictionary will be data dependent but its construction is not part of the above optimization

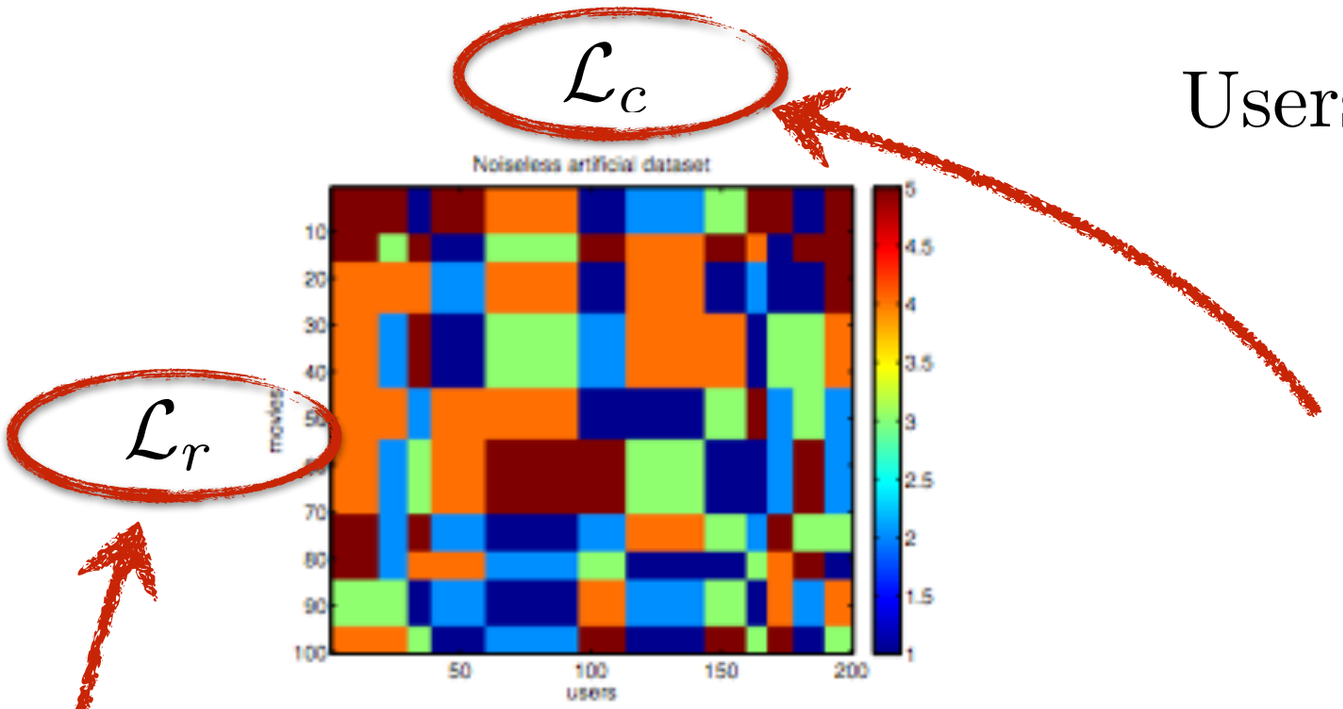
Learning on with Graphs



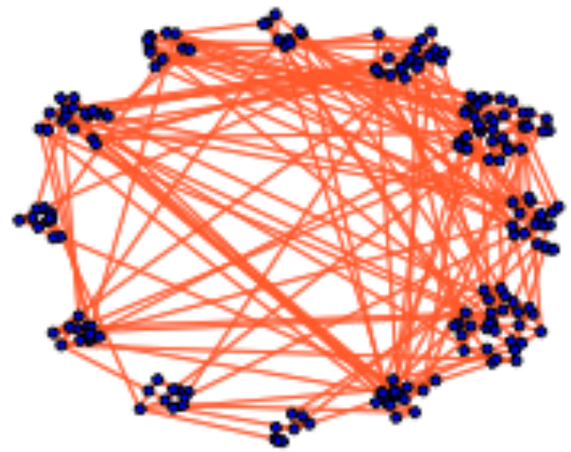




Application: A Recommender System



Users structured as *communities*



$\boldsymbol{X}$[movie, user] = movie rating

Users in community rate similarly

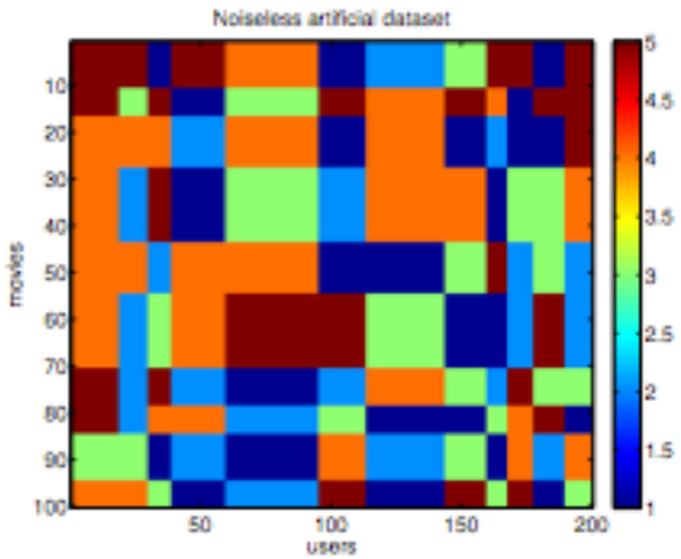
Movies are clustered in genres.

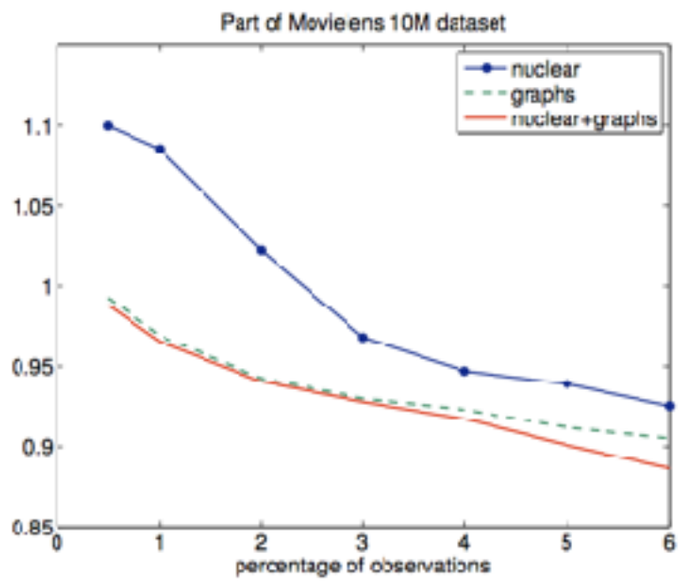
Similar movies are rated similarly by users

Application: A Recommender System

$\mathcal{L}_c$

$\mathcal{L}_r$





$\boldsymbol{X}[\text{movie, user}] = \text{movie rating}$

$$\arg\min_{\mathbf{X}} \|A_\Omega \circ (\mathbf{X} - \mathbf{M})\| + \gamma_r \mathrm{tr}\left(\mathbf{X}\mathbf{L}_r\mathbf{X}^t\right) + \gamma_c\left(\mathrm{tr}\mathbf{X}^t\mathbf{L}_c\mathbf{X}\right)$$