




GSP for ML



PREVIOUSLY
ON...

SP on Graphs Cheat Sheet

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

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

Laplacian



$$-\Delta$$

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



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

Graph Fourier

Frequencies

Filter and Filtering

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

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



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

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



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

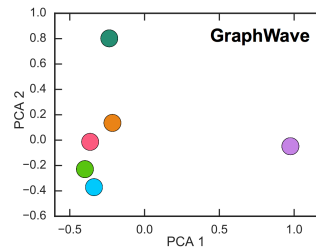
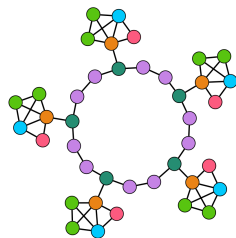
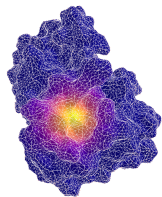
Previously on GSP

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

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}_{ν_1} of underlying feature map

 Interpolate to recover missing features (transductive learning)

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

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

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

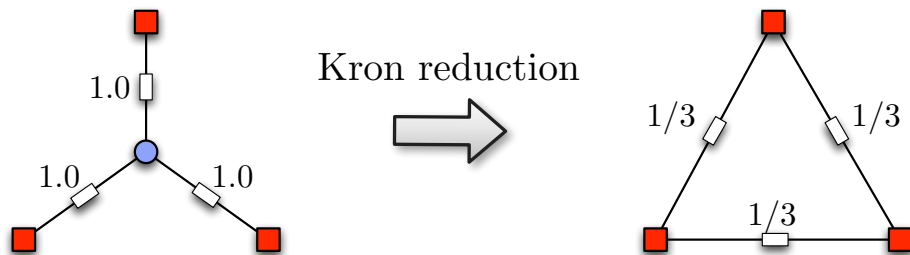
Coarsening by Kron Reduction

In order to iterate the construction, we need to construct a graph on the reduced vertex set.

$$\mathbf{A}_r = \mathbf{A}[\alpha, \alpha] - \mathbf{A}[\alpha, \alpha) \mathbf{A}(\alpha, \alpha)^{-1} \mathbf{A}(\alpha, \alpha]$$

Schur complement

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}[\alpha, \alpha] & \mathbf{A}[\alpha, \alpha) \\ \mathbf{A}(\alpha, \alpha] & \mathbf{A}(\alpha, \alpha) \end{bmatrix}$$



Dorfler et al., ArXiv, 2011

A simple interpolation example

Graph filtering interpretation of the algorithm:

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

Kron-reduced regularised Laplacian = Laplacian of Kron-reduced graph

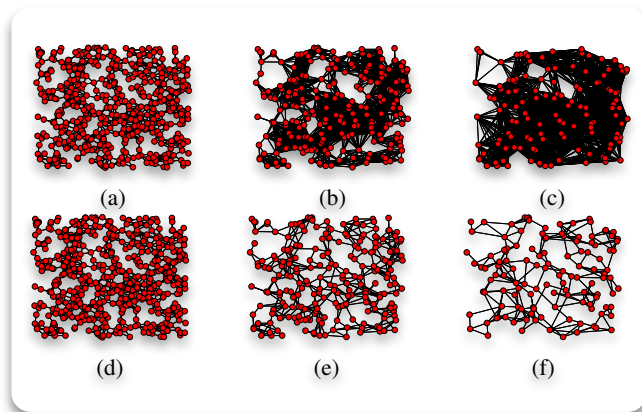
Upsample coefficients as graph signal (set of weighted kroneckers):

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

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

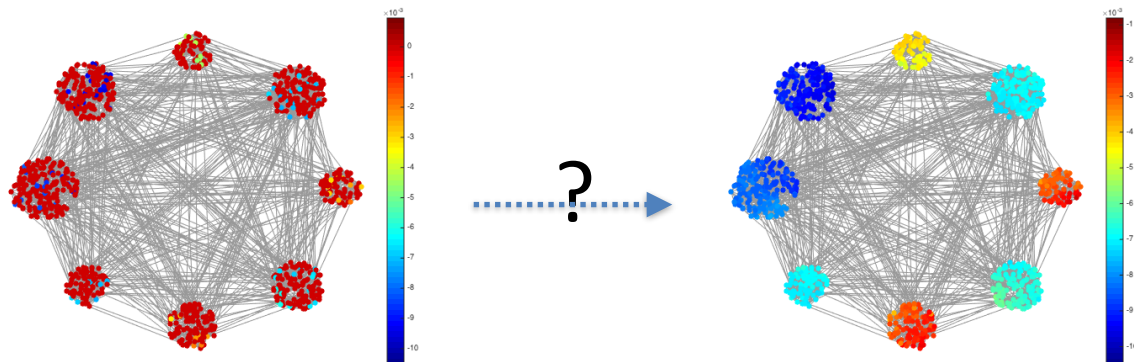


Back to Spectral Clustering

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

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

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)

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

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

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

Sampling Model

$$\frac{\|\mathbf{U}_k^\top \boldsymbol{\delta}_i\|_2}{\|\mathbf{U}^\top \boldsymbol{\delta}_i\|_2} = \frac{\|\mathbf{U}_k^\top \boldsymbol{\delta}_i\|_2}{\|\boldsymbol{\delta}_i\|_2} = \|\mathbf{U}_k^\top \boldsymbol{\delta}_i\|_2$$

How much a perfect impulse can be concentrated on first k eigenvectors

Carries interesting information about the graph

Ideally: \mathbf{p}_i large wherever $\|\mathbf{U}_k^\top \boldsymbol{\delta}_i\|_2$ is large

Graph Coherence

$$\nu_{\mathbf{p}}^k := \max_{1 \leq i \leq n} \left\{ \mathbf{p}_i^{-1/2} \|\mathbf{U}_k^\top \boldsymbol{\delta}_i\|_2 \right\}$$

$$\text{Rem: } \nu_{\mathbf{p}}^k \geq \sqrt{k}$$

Stable Embedding

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

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

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

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

$$\mathbf{M} \mathbf{P}^{-1/2} \mathbf{x} = \mathbf{P}_{\Omega}^{-1/2} \mathbf{M} \mathbf{x} \quad \text{Only need } \mathbf{M}, \text{ re-weighting offline}$$

$$(\nu_{\mathbf{p}}^k)^2 \geq k \quad \text{Need to sample at least } k \text{ nodes}$$

Proof similar to CS in bounded ONB but simpler since model is a subspace (not a union)

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

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

$\nu_k = \sqrt{N} \max_{1 \leq i \leq N} \{\|U_k^T \delta_i\|_2\} \quad \sqrt{k} \leq \nu_k \leq \sqrt{N}$



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)

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

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

Standard Decoder

$$\min_{\mathbf{z} \in \text{span}(\mathbf{U}_k)} \left\| \mathbf{P}_{\Omega}^{-1/2} (\mathbf{M}\mathbf{z} - \mathbf{y}) \right\|_2$$

need projector

re-weighting for RIP

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

Estimating the Optimal Distribution

Need to estimate $\|\mathbf{U}_k^\top \boldsymbol{\delta}_i\|_2^2$

Filter random signals with ideal low-pass filter:

$$\mathbf{r}_{b_{\lambda_k}} = \mathbf{U} \operatorname{diag}(\boldsymbol{\lambda}_1, \dots, \boldsymbol{\lambda}_k, 0, \dots, 0) \mathbf{U}^\top \mathbf{r} = \mathbf{U}_k \mathbf{U}_k^\top \mathbf{r}$$

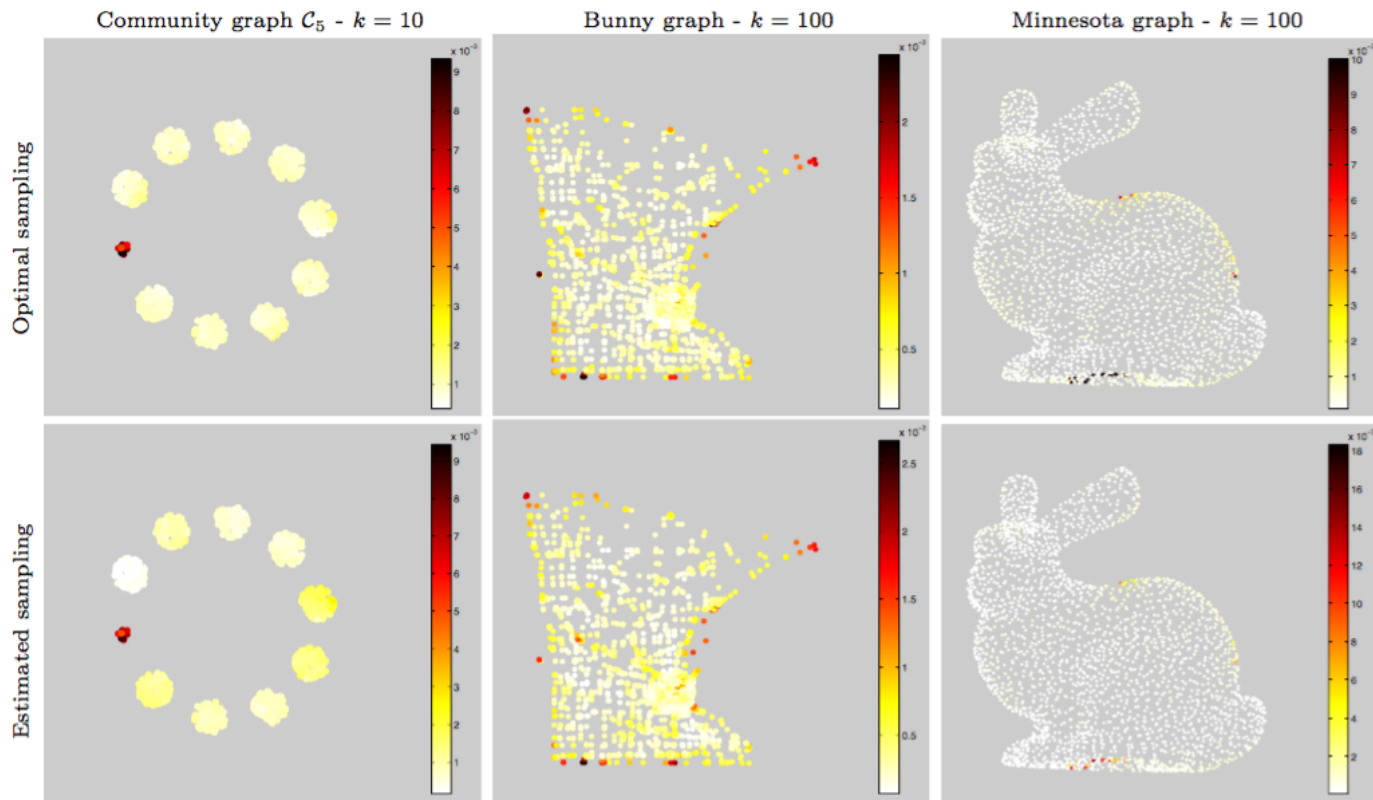
$$\mathbb{E}(\mathbf{r}_{b_{\lambda_k}})_i^2 = \boldsymbol{\delta}_i^\top \mathbf{U}_k \mathbf{U}_k^\top \mathbb{E}(\mathbf{r} \mathbf{r}^\top) \mathbf{U}_k \mathbf{U}_k^\top \boldsymbol{\delta}_i = \|\mathbf{U}_k^\top \boldsymbol{\delta}_i\|_2^2$$

In practice, one may use a polynomial approximation of the ideal filter and:

$$\tilde{\mathbf{p}}_i := \frac{\sum_{l=1}^L (\mathbf{r}_{c_{\lambda_k}}^l)_i^2}{\sum_{i=1}^n \sum_{l=1}^L (\mathbf{r}_{c_{\lambda_k}}^l)_i^2}$$

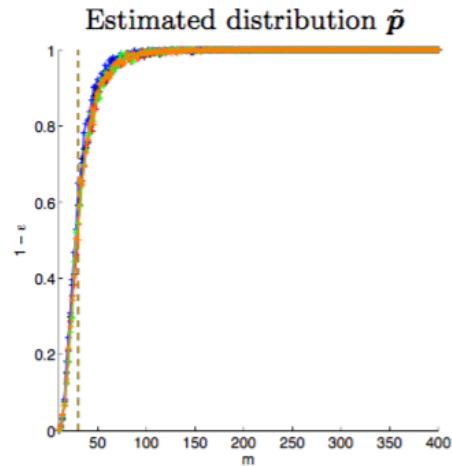
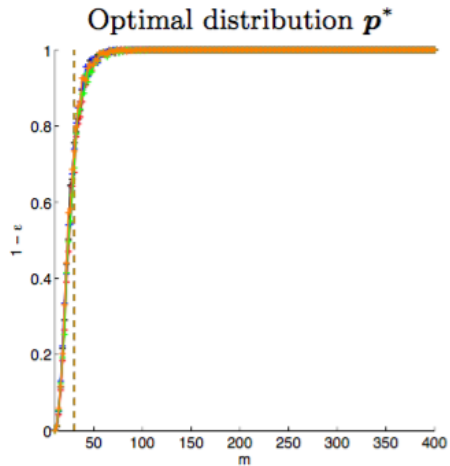
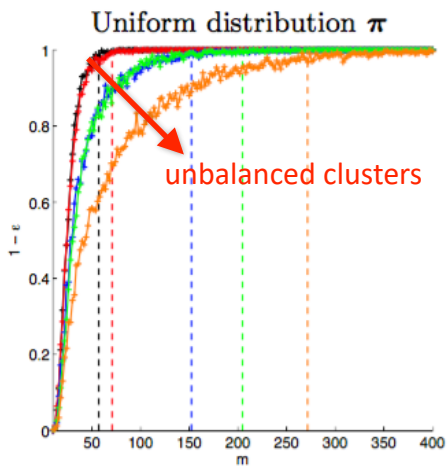
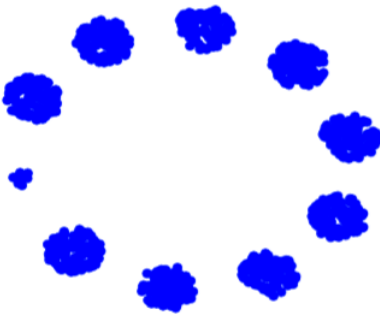
$$L \geq \frac{C}{\delta^2} \log \left(\frac{2n}{\epsilon} \right)$$

Experiments

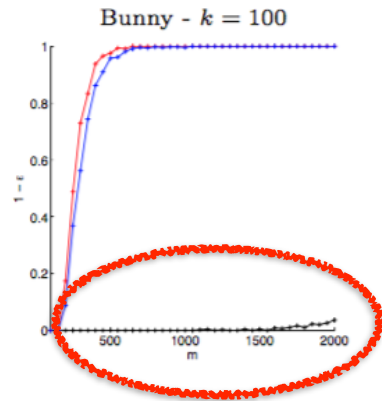
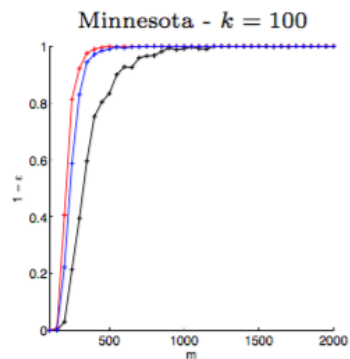
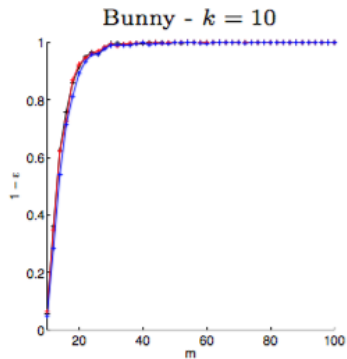
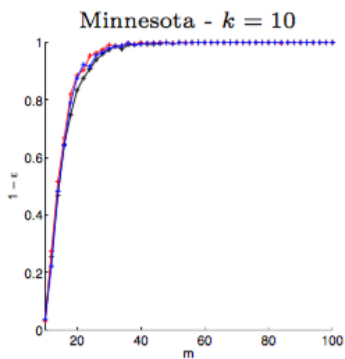
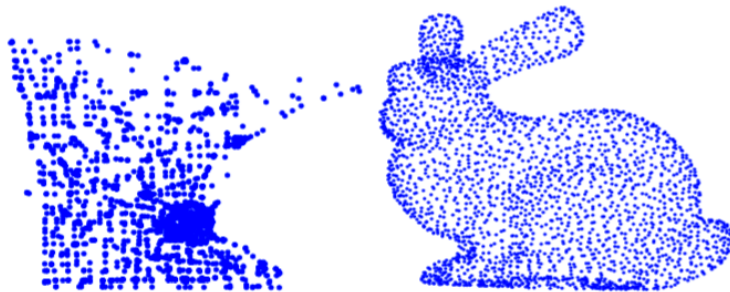


Experiments

Community graph



Experiments



Experiments

Original



Reconstructed (sampling with \tilde{p})



Reconstructed (sampling with π)

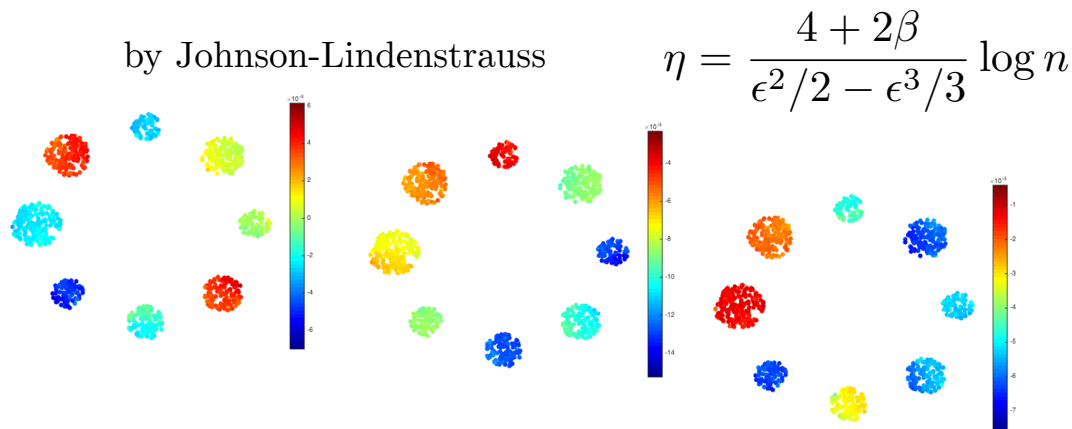


Compressive Spectral Clustering

Clustering equivalent to recovery of cluster assignment functions

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

Generate features by filtering random signals



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

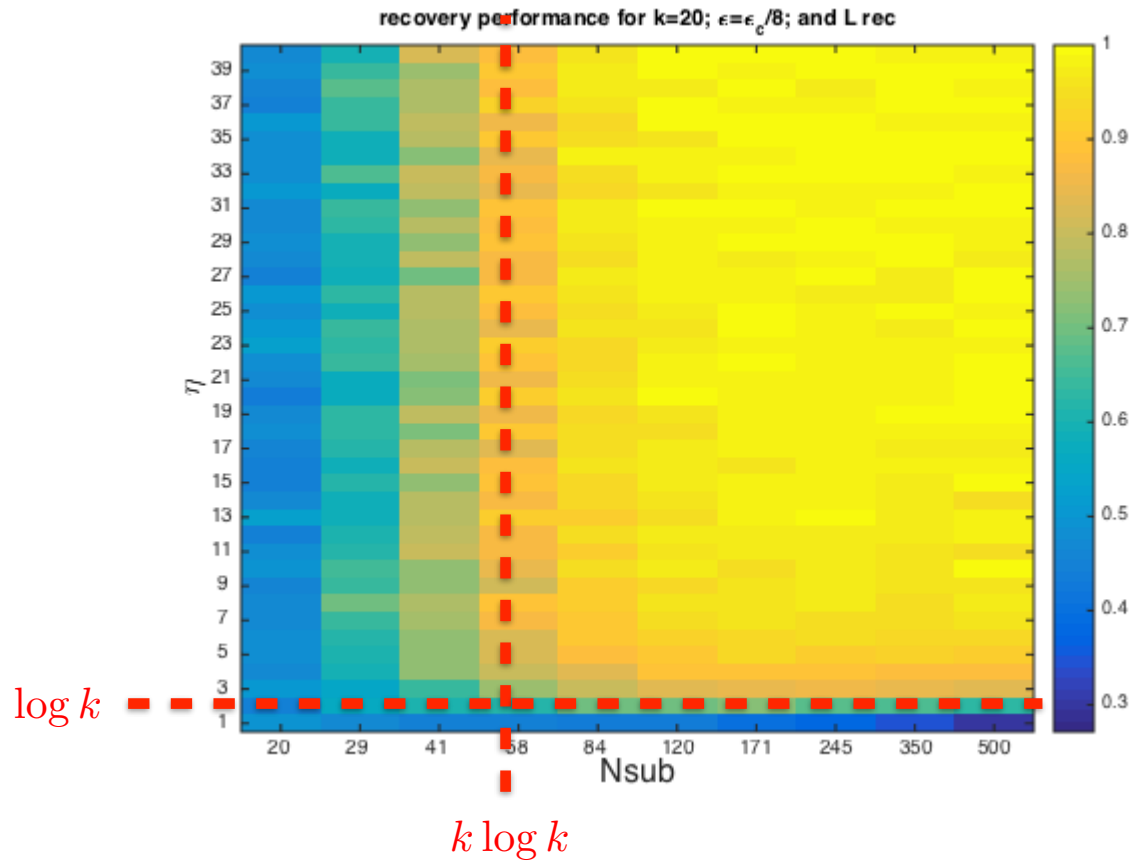
by Johnson-Lindenstrauss
$$\eta = \frac{4 + 2\beta}{\epsilon^2/2 - \epsilon^3/3} \log n$$

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

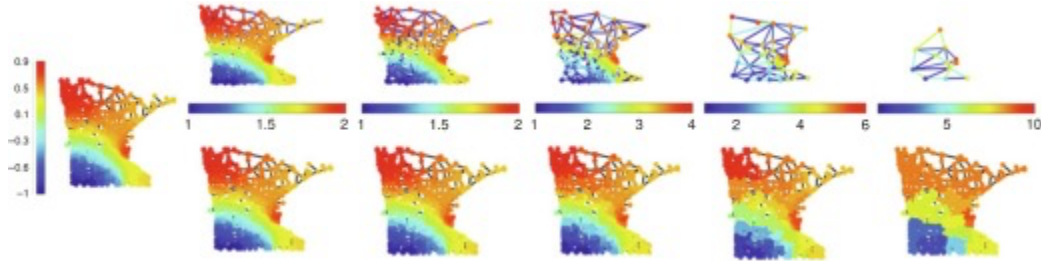
Compressive Spectral Clustering



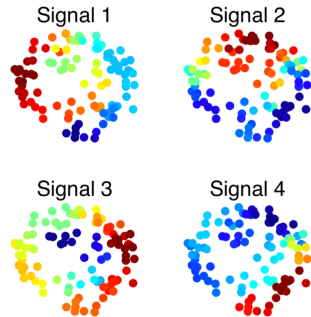
Other developments

25

Filterbanks, wavelets and other transforms on graphs



Stationary graph signal processing
(statistical “shift” invariance)



but also sampling, compression ...

Pattern Recognition and Machine Learning (in one slide)

Schematically Machine Learning seeks to build a map from data x to decisions y

$y = \mathcal{F}(x)$ the output decision can be numerical, categorical, ...

Not so long ago, this involved decomposing

$$\mathcal{F} = \boxed{\mathcal{F}_{\text{decision}}} \circ \boxed{\mathcal{F}_{\text{features}}}$$

learnt from data hand-crafted using domain knowledge

The modern version (“AI revival”)

$$\mathcal{F} = \boxed{\mathcal{F}_{\text{decision}}} \circ \boxed{\mathcal{F}_{\text{layer } n} \circ \dots \circ \mathcal{F}_{\text{layer } 1}}$$

learnt from data !

Neural Nets

A (feed-forward) Neural Net is one possible feature extraction layer

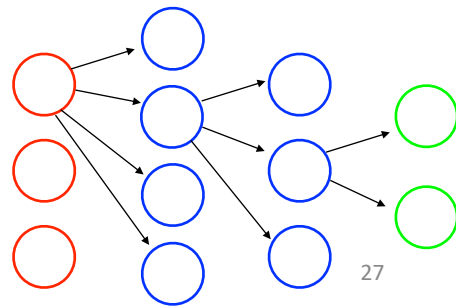
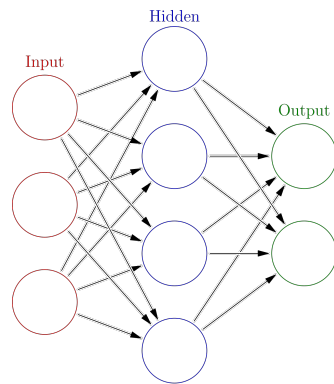
$$\mathcal{F}_{i^{\text{th}} \text{ neuron}}(x) = \sigma(w_i^T x + b_i)$$

weighted average of input
+ bias/offset

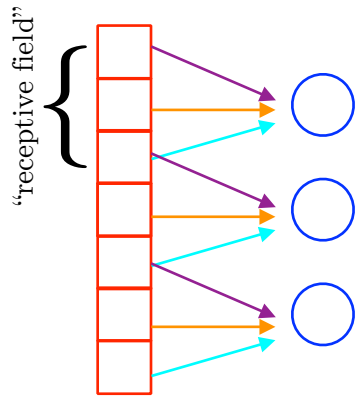
non-linear activation

$$\mathcal{F}_{\text{neural layer}}(x) = \sigma(Wx + b)$$

NN layers can be stacked,
with different parameters at each layer
= a multilayer (or deep) Neural Network



Convolutional Neural Nets



The same set of weights is re-used in a translation invariant way

$$\mathcal{F}_{\text{conv layer}}(x) = \sigma(w * x + b)$$

Motivations: Inductive bias for images, sound (translation invariance, ...)

works because input data is structured on regular grid

Computational efficiency (apply weights in effective manner)

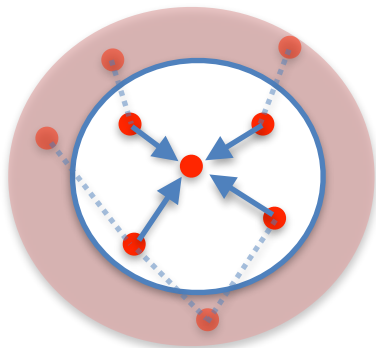
Learning efficiency (weight sharing, $O(1)$ parameters per layer)

Graph Neural Nets (GNNs)

Reminder: $\mathcal{F}_{\text{neural layer}}(x) = \sigma(Wx + b)$

Idea 1: work at the node level to compute local aggregators (permutation invariance)
node features, edge features \rightarrow new node features
output: node-level representations, graph-level representation

Idea 2: make computations scalable by weight sharing (“convolutional net”)



$$\mathcal{F}_{i^{\text{th}} \text{ node}}(x) = \sigma(w_i^T x + b_i)$$

w_i \rightarrow neighbourhood of node i = “receptive field”
 \rightarrow should be independent of neighbourhood ???

Convolutional GNNs

ChebNet - truly scalable GNNs

Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering, NeurIPS'16

Generate GNNs that parameterised spectral graph filters

$$W = U D_{\Theta} U^T \quad \# \text{nodes shared free parameters } \Theta$$

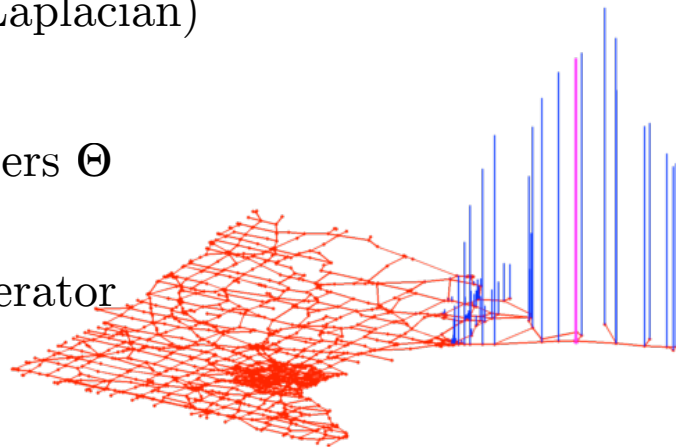
U = eigenvectors of shift operator (Laplacian)

$$W = U G_{\Theta}(\Lambda) U^T \quad \text{Parametric form with } O(1) \text{ parameters } \Theta$$

$$W = P_{\Theta}(L) \quad \text{Special form, polynomial of shift operator}$$

$$\mathcal{F}_{i^{\text{th}} \text{ node}}(x) = \sigma(\mathbf{w}_i^T \mathbf{x} + b_i) \quad \mathbf{w}_i = [P_{\Theta}(L)]_i$$

same parameters for all nodes



Convolutional GNNs

GCN - Simplified architecture [Kipf & Welling, ICLR'17]

Key simplifications: Set $\lambda_{\max} = 2$ so that $\bar{L} = \frac{2L}{\lambda_{\max}} - \mathbb{I} = -A$

Use only linear filters ($K=1$) $W = P_{\Theta}(A) = \underbrace{\theta^{(0)}}_{\text{node}} \mathbb{I} - \underbrace{\theta^{(1)}}_{\text{1-hop neighbours}} A$

Use same weights for nodes and their direct neighbours

$$\begin{aligned}\theta = \theta^{(0)} = -\theta^{(1)} &\Rightarrow \mathcal{F}_{\text{GCN}}(x) = \sigma(Wx) \\ &= \sigma(\theta(\mathbb{I} + A)x) \\ &= \sigma(\theta \tilde{A}x)\end{aligned}$$

Re-scale for stability $\tilde{A} \rightarrow D^{-1/2} \tilde{A} D^{-1/2}$

$$\mathcal{F}_{\text{GCN}}(x) = \sigma(\theta D^{-1/2}(\mathbb{I} + A)D^{-1/2}x)$$

Message Passing GNNs

Reminder: The Graph Conv Layer

$$W = P_{\Theta}(L) \quad \mathcal{F}_{i^{\text{th}} \text{ node}}(x) = \sigma(\textcolor{blue}{w}_i^T \textcolor{red}{x} + b_i) \quad \textcolor{blue}{w}_i = [P_{\Theta}(L)]_i$$

$$P_{\Theta}(L) = \sum_{k=0}^{K-1} \Theta_k T_k(\bar{L}), \text{ where } \bar{L} = 2L/\lambda_{\max} - \mathbb{I}$$

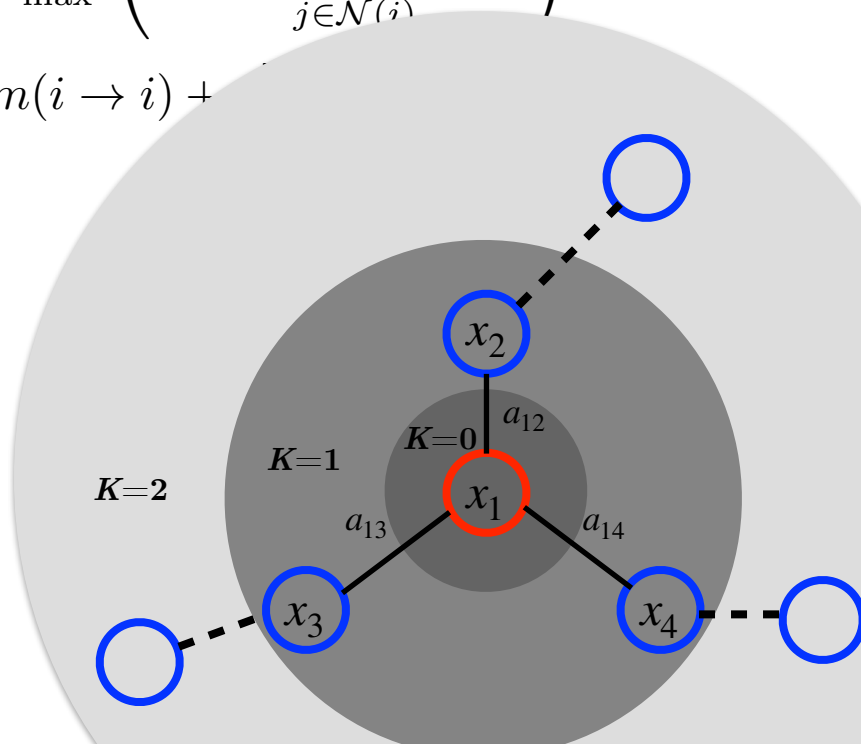
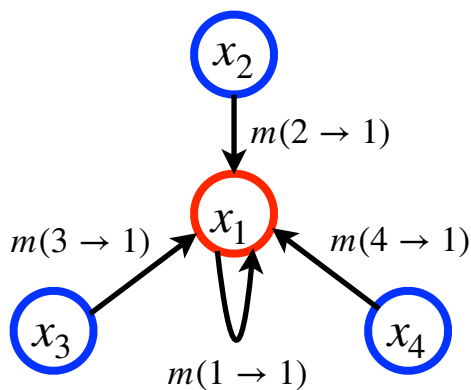
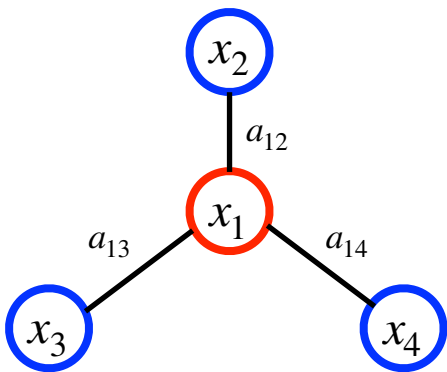
Chebyshev polynomials: $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$, $T_0(x) = 1$, $T_1(x) = x$

Therefore: $Wx = [P_{\Theta}(L)]x$

$$= \sum_{k=0}^{K-1} \Theta_k \textcolor{blue}{\boxed{T_k(\bar{L})x}} \textcolor{blue}{\text{recursively passing messages between nodes}}$$

Message Passing GNNs

$$\begin{aligned}
 Wx &= [P_{\Theta}(L)]x \\
 &= \sum_{k=0}^{K-1} \Theta_k T_k(\bar{L})x
 \end{aligned}
 \qquad
 \begin{aligned}
 (\bar{L}x)_i &= \frac{2}{\lambda_{\max}} \left(d_i x_i - \sum_{j \in \mathcal{N}(i)} a_{ij} x_j \right) - x_i \\
 &= m(i \rightarrow i) + \dots
 \end{aligned}$$



No Graphs ? No Problem !

Sometimes the graph is a proxy for the “manifold hypotheses”:

Features span a low dimensional manifold



But often a more accurate statement is:

There are structured high-density regions in feature space

No Graphs ?

Kernel Operators

Consider \mathbb{R}^d endowed with the probability distribution $p(x)$.

\mathcal{H}_p the corresponding Hilbert space:

$$\langle f, g \rangle_{\mathcal{H}_p} = \int_{\mathbb{R}^d} p(y) dy f^*(y) g(y).$$

Let $k : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ be a Mercer kernel

$$T_k f(x) = \int_{\mathbb{R}^d} p(y) dy k(x, y) f(y) \quad T_k u_i(x) = \sigma_i u_i \Rightarrow \hat{f}[i] = \langle \phi_i, f \rangle_{\mathcal{H}_p}$$

$$T_k f(x) = \sum_i u_i(x) \left\{ \sigma_i \hat{f}[i] \right\} \quad g(T_k) f(x) = \sum_i u_i(x) \left\{ g(\sigma_i) \hat{f}[i] \right\}$$

Sketching kernels for density filters

$$\mathcal{Z}(x) = \frac{1}{\sqrt{D}} [z_{\omega_1}(x), \dots, z_{\omega_D}(x)], \omega_k \sim \text{i.i.d. } \hat{k}(\omega) \quad \text{where } z_{\omega}(x) = e^{j\omega^T x}$$

$$\mathcal{Z}^H(x)\mathcal{Z}(y) = \frac{1}{D} \sum_{k=1}^D \bar{z}_{\omega_k}(x) z_{\omega_k}(y) \simeq k(x-y)$$

$$\text{RFF matrix } \mathcal{Z} = [\mathcal{Z}(x_1), \dots, \mathcal{Z}(x_N)] \in \mathbb{R}^{D \times N} \quad \mathcal{Z}^H \mathcal{Z} \simeq K$$

$$M = \mathcal{Z} \mathcal{Z}^H \in \mathbb{R}^{D \times D} \quad g(\tilde{K}) = \mathcal{Z}^H h_g(M) \mathcal{Z}$$

Note that RFF is like a neural net: Linear layer (random weights), point wise non-lin.

A funny point of view ...

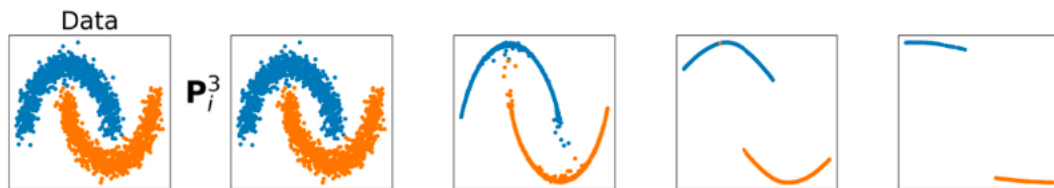
$$g(\tilde{K}) = \mathcal{Z}^H h_g(M) \mathcal{Z}$$

RFF can be seen as a “data driven” transformation

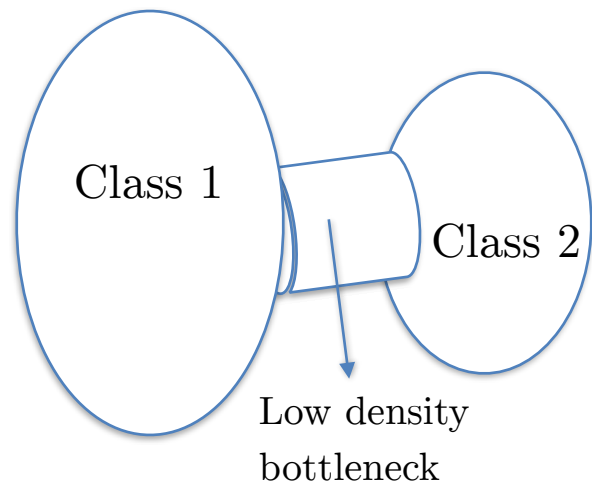
It maps to a data/kernel driven “Fourier” domain

“Fourier coefficients” of the data can be manipulated

Example: **density condensation**, apply filter to data coordinates, update density



Unsupervised, semi-supervised, supervised ML revisited



Label function **must** be smooth over data
(Dirichlet kernel semi-norm)

$$\max_F S_D\{F\} = \max_F F^T K_D F = \max_F \|Z_D F\|^2$$

Just like spectral clustering, but defined at
any data point (via the kernel)

Reverse perspective: select a parametric feature extractor $Z_\theta(x)$ (neural net, ...)

Implicit kernel: $K_\theta = Z_\theta Z_\theta^T$

With labeled data $(x_i, F(x_i))$: $\max_\theta \|Z_\theta F\|^2$