Wavelets on Graphs, an Introduction

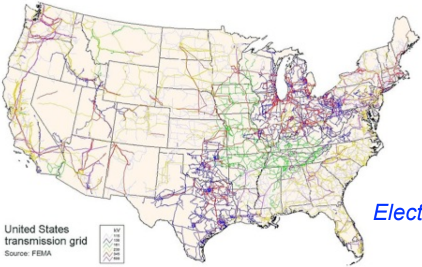
Pierre Vandergheynst and David Shuman

Ecole Polytechnique Fédérale de Lausanne (EPFL)
Signal Processing Laboratory
{pierre.vandergheynst,david.shuman}@epfl.ch

Université de Provence
Marseille, France

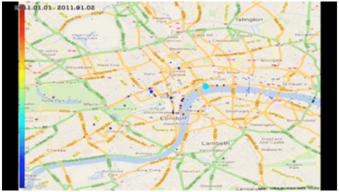
November 17, 2011

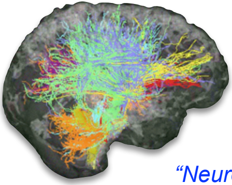
Processing Signals on Graphs



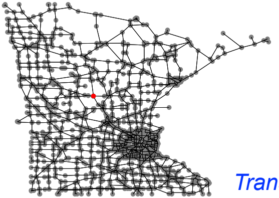
Electrical Network

Social Network





"Neuronal" Network



Transportation Network

Outline

1. Introduction
2. Spectral Graph Theory Background
   - Definitions
   - Differential Operators on Graphs
   - Graph Laplacian Eigenvectors
   - Two Applications of Graph Laplacian Eigenvectors
   - Graph Downsampling
   - Filtering on Graphs
3. Wavelet Constructions on Graphs
4. Approximate Graph Multiplier Operators
5. Distributed Signal Processing via the Chebyshev Approximation
6. Open Issues and Challenges

Spectral Graph Theory Notation

- Connected, undirected, weighted graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, W\}$

- Degree matrix D: zeros except diagonals, which are sums of weights of edges incident to corresponding node

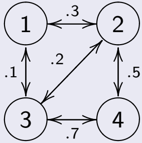
- Non-normalized Laplacian: $\mathcal{L} := D - W$

- Complete set of orthonormal eigenvectors and associated real, non-negative eigenvalues:

$$\mathcal{L}\chi_\ell = \lambda_\ell \chi_\ell,$$

ordered w.l.o.g. s.t.

$$0 = \lambda_0 < \lambda_1 \le \lambda_2 ... \le \lambda_{N-1} := \lambda_{\max}$$

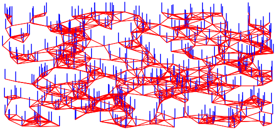


$$W = \begin{bmatrix} 0 & .3 & .1 & 0 \\ .3 & 0 & .2 & .5 \\ .1 & .2 & 0 & .7 \\ 0 & .5 & .7 & 0 \end{bmatrix}$$

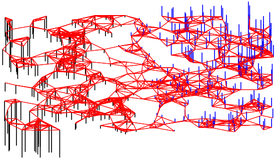
$$D = \begin{bmatrix} .4 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1.2 \end{bmatrix}$$

Graph Laplacian Eigenvectors

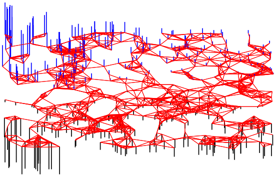
- Values of eigenvectors associated with lower frequencies (low λ_ℓ) change less rapidly across connected vertices



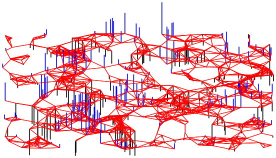
χ_0



χ_1



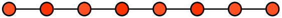
χ_2



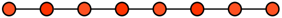
χ_{50}

Graph Laplacian Eigenvectors

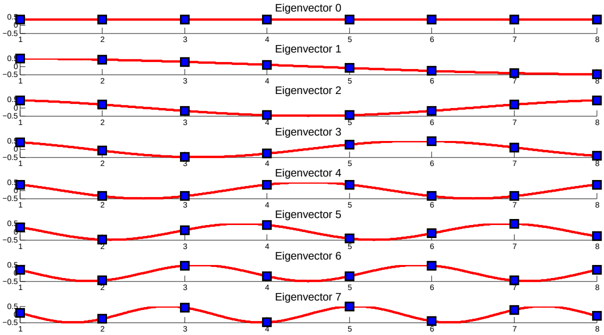
Special Case – Path Graph



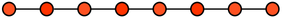
Graph Laplacian Eigenvectors
Special Case – Path Graph



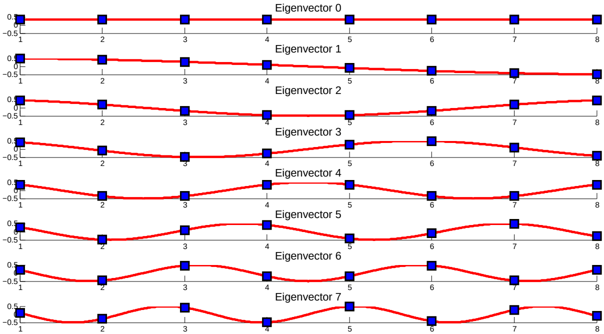
- $\lambda_\ell = 2 - 2\cos\left(\frac{\pi\ell}{N}\right)$
- $\chi_0(i) = \frac{1}{\sqrt{N}},\ \chi_\ell(i) = \sqrt{\frac{2}{N}}\cos\left(\frac{\pi\ell(i-0.5)}{N}\right),\ \ell = 1, 2, \ldots, N-1$



Graph Laplacian Eigenvectors
Special Case – Path Graph

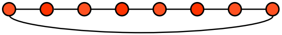


- $\lambda_\ell = 2 - 2\cos\left(\frac{\pi\ell}{N}\right)$
- $\chi_0(i) = \frac{1}{\sqrt{N}},\ \chi_\ell(i) = \sqrt{\frac{2}{N}}\cos\left(\frac{\pi\ell(i-0.5)}{N}\right),\ \ell = 1, 2, \ldots, N-1$

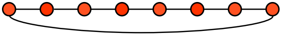


$\begin{bmatrix} | & & | \\ \chi_0 & \cdots & \chi_{N-1} \\ | & & | \end{bmatrix}$ is the Discrete Cosine Transform matrix (DCT-II, Strang, 1999), which is used in JPEG image compression

Graph Laplacian Eigenvectors
Special Case – Ring Graph

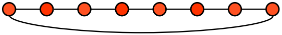


Graph Laplacian Eigenvectors
Special Case – Ring Graph



- (Unordered) Laplacian eigenvalues: $\lambda_\ell = 2 - 2\cos\left(\frac{2\ell\pi}{N}\right)$

Graph Laplacian Eigenvectors
Special Case – Ring Graph



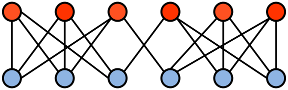
- (Unordered) Laplacian eigenvalues: $\lambda_\ell = 2 - 2\cos\left(\frac{2\ell\pi}{N}\right)$

- One possible choice of orthogonal Laplacian eigenvectors:

$$\chi_\ell = \left[1, \omega^\ell, \omega^{2\ell}, \ldots, \omega^{(N-1)\ell}\right], \text{ where } \omega = e^{\frac{2\pi j}{N}}$$

- $\begin{bmatrix} | & & | \\ \chi_0 & \cdots & \chi_{N-1} \\ | & & | \end{bmatrix}$ is the Discrete Fourier Transform (DFT) matrix

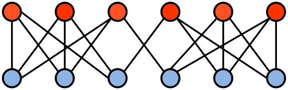
Graph Laplacian Eigenvectors
Special Case – k-Regular Bipartite Graphs



- A graph $\mathcal{G}$ is *bipartite* if $\mathcal{V}$ can be partitioned into subsets $\mathcal{V}_1$ and $\mathcal{V}_1^c$ so that every edge $e \in \mathcal{E}$ connects one vertex in $\mathcal{V}_1$ with one vertex in $\mathcal{V}_1^c$

Graph Laplacian Eigenvectors

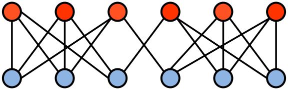
Special Case – k-Regular Bipartite Graphs



- A graph $\mathcal{G}$ is *bipartite* if $\mathcal{V}$ can be partitioned into subsets $\mathcal{V}_1$ and $\mathcal{V}_1^c$ so that every edge $e \in \mathcal{E}$ connects one vertex in $\mathcal{V}_1$ with one vertex in $\mathcal{V}_1^c$

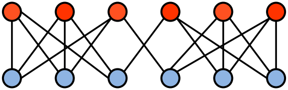
- A graph $\mathcal{G}$ is *k-regular* if every vertex has the same degree (k)

Graph Laplacian Eigenvectors
Special Case – k-Regular Bipartite Graphs



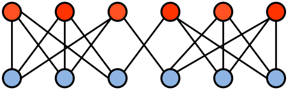
- A graph $\mathcal{G}$ is *bipartite* if $\mathcal{V}$ can be partitioned into subsets $\mathcal{V}_1$ and $\mathcal{V}_1^c$ so that every edge $e \in \mathcal{E}$ connects one vertex in $\mathcal{V}_1$ with one vertex in $\mathcal{V}_1^c$
- A graph $\mathcal{G}$ is *k-regular* if every vertex has the same degree (k)
- All k-regular bipartite graphs have an even number N of vertices, and $\mathcal{V}_1$ has $\frac{N}{2}$ vertices

Graph Laplacian Eigenvectors
Special Case – k-Regular Bipartite Graphs



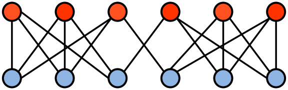
- A graph $\mathcal{G}$ is *bipartite* if $\mathcal{V}$ can be partitioned into subsets $\mathcal{V}_1$ and $\mathcal{V}_1^c$ so that every edge $e \in \mathcal{E}$ connects one vertex in $\mathcal{V}_1$ with one vertex in $\mathcal{V}_1^c$
- A graph $\mathcal{G}$ is *k-regular* if every vertex has the same degree (k)
- All k-regular bipartite graphs have an even number N of vertices, and $\mathcal{V}_1$ has $\frac{N}{2}$ vertices
- Laplacian eigenvalues satisfy $\lambda_\ell = 2k - \lambda_{N-1-\ell}$

Graph Laplacian Eigenvectors
Special Case – k-Regular Bipartite Graphs



- A graph $\mathcal{G}$ is *bipartite* if $\mathcal{V}$ can be partitioned into subsets $\mathcal{V}_1$ and $\mathcal{V}_1^c$ so that every edge $e \in \mathcal{E}$ connects one vertex in $\mathcal{V}_1$ with one vertex in $\mathcal{V}_1^c$
- A graph $\mathcal{G}$ is *k-regular* if every vertex has the same degree (k)
- All k-regular bipartite graphs have an even number N of vertices, and $\mathcal{V}_1$ has $\frac{N}{2}$ vertices
- Laplacian eigenvalues satisfy $\lambda_\ell = 2k - \lambda_{N-1-\ell}$
- If $\chi_\ell = \begin{bmatrix} \chi_\ell^1 \\ \\ \chi_\ell^{1c} \end{bmatrix}$, then $\chi_{N-1-\ell} = \begin{bmatrix} \chi_\ell^1 \\ \\ -\chi_\ell^{1c} \end{bmatrix}$

Graph Laplacian Eigenvectors
Special Case – k-Regular Bipartite Graphs



- A graph $\mathcal{G}$ is *bipartite* if $\mathcal{V}$ can be partitioned into subsets $\mathcal{V}_1$ and $\mathcal{V}_1^c$ so that every edge $e \in \mathcal{E}$ connects one vertex in $\mathcal{V}_1$ with one vertex in $\mathcal{V}_1^c$
- A graph $\mathcal{G}$ is *k-regular* if every vertex has the same degree (k)
- All k-regular bipartite graphs have an even number N of vertices, and $\mathcal{V}_1$ has $\frac{N}{2}$ vertices
- Laplacian eigenvalues satisfy $\lambda_\ell = 2k - \lambda_{N-1-\ell}$
- If $\chi_\ell = \begin{bmatrix} \chi_\ell^1 \\ \\ \chi_\ell^{1c} \end{bmatrix}$, then $\chi_{N-1-\ell} = \begin{bmatrix} \chi_\ell^1 \\ \\ -\chi_\ell^{1c} \end{bmatrix}$
- For $\mathcal{L}^{norm}$, $\lambda_\ell = 2 - \lambda_{N-1-\ell}$ and the Laplacian eigenvector property holds for any (non-regular) bipartite graph as well

Outline

1 Introduction

2 Spectral Graph Theory Background

- Definitions
- Differential Operators on Graphs
- Graph Laplacian Eigenvectors
- Two Applications of Graph Laplacian Eigenvectors
- Graph Downsampling
- Filtering on Graphs

3 Wavelet Constructions on Graphs

4 Approximate Graph Multiplier Operators

5 Distributed Signal Processing via the Chebyshev Approximation

6 Open Issues and Challenges

Spectral Clustering

- Goal: Partition the graph into k roughly equal-sized clusters such that the edges between different clusters have low weights

Spectral Clustering

- Goal: Partition the graph into k roughly equal-sized clusters such that the edges between different clusters have low weights
- $\text{cut}(\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_k) := \frac{1}{2}\sum_{i=1}^{k} W(\mathcal{V}_i, \mathcal{V}_i^c)$

Spectral Clustering

- Goal: Partition the graph into k roughly equal-sized clusters such that the edges between different clusters have low weights
- $\text{cut}(\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_k) := \frac{1}{2} \sum_{i=1}^{k} W(\mathcal{V}_i, \mathcal{V}_i^c)$
- To encourage balanced cluster sizes, minimize, e.g.,

$$\text{RatioCut}(\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_k) := \frac{1}{2} \sum_{i=1}^{k} \frac{W(\mathcal{V}_i, \mathcal{V}_i^c)}{|\mathcal{V}_i|}$$

Spectral Clustering

- Goal: Partition the graph into k roughly equal-sized clusters such that the edges between different clusters have low weights
- $\text{cut}(\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_k) := \frac{1}{2}\sum_{i=1}^{k} W(\mathcal{V}_i, \mathcal{V}_i^c)$
- To encourage balanced cluster sizes, minimize, e.g.,

$$\text{RatioCut}(\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_k) := \frac{1}{2}\sum_{i=1}^{k} \frac{W(\mathcal{V}_i, \mathcal{V}_i^c)}{|\mathcal{V}_i|}$$

<u>Example: $k = 2$</u> (von Luxburg, 2007)

- For a fixed subset $\mathcal{V}_1 \subset \mathcal{V}$, define $f \in \mathbb{R}^N$ by $f_i := \begin{cases} \sqrt{\frac{|\mathcal{V}_1^c|}{|\mathcal{V}_1|}} & \text{, if } i \in \mathcal{V}_1 \\ -\sqrt{\frac{|\mathcal{V}_1|}{|\mathcal{V}_1^c|}} & \text{, if } i \in \mathcal{V}_1^c \end{cases}$

- $\|f\|_2^2 = |\mathcal{V}_1|\frac{|\mathcal{V}_1^c|}{|\mathcal{V}_1|} + |\mathcal{V}_1^c|\frac{|\mathcal{V}_1|}{|\mathcal{V}_1^c|} = N$
- $\sum_{i=1}^{N} f_i = |\mathcal{V}_1|\sqrt{\frac{|\mathcal{V}_1^c|}{|\mathcal{V}_1|}} - |\mathcal{V}_1^c|\sqrt{\frac{|\mathcal{V}_1|}{|\mathcal{V}_1^c|}} = 0$

Spectral Clustering

- Goal: Partition the graph into k roughly equal-sized clusters such that the edges between different clusters have low weights
- $\text{cut}(\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_k) := \frac{1}{2}\sum_{i=1}^{k} W(\mathcal{V}_i, \mathcal{V}_i^c)$
- To encourage balanced cluster sizes, minimize, e.g.,

$$\text{RatioCut}(\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_k) := \frac{1}{2}\sum_{i=1}^{k} \frac{W(\mathcal{V}_i, \mathcal{V}_i^c)}{|\mathcal{V}_i|}$$

<u>EXAMPLE: $k = 2$</u> (von Luxburg, 2007)

- For a fixed subset $\mathcal{V}_1 \subset \mathcal{V}$, define $f \in \mathbb{R}^N$ by $f_i := \begin{cases} \sqrt{\frac{|\mathcal{V}_1^c|}{|\mathcal{V}_1|}} & \text{, if } i \in \mathcal{V}_1 \\ -\sqrt{\frac{|\mathcal{V}_1|}{|\mathcal{V}_1^c|}} & \text{, if } i \in \mathcal{V}_1^c \end{cases}$
- $\|f\|_2^2 = |\mathcal{V}_1|\frac{|\mathcal{V}_1^c|}{|\mathcal{V}_1|} + |\mathcal{V}_1^c|\frac{|\mathcal{V}_1|}{|\mathcal{V}_1^c|} = N$
- $\sum_{i=1}^{N} f_i = |\mathcal{V}_1|\sqrt{\frac{|\mathcal{V}_1^c|}{|\mathcal{V}_1|}} - |\mathcal{V}_1^c|\sqrt{\frac{|\mathcal{V}_1|}{|\mathcal{V}_1^c|}} = 0$
-

$$\begin{aligned} f^{\mathrm{T}}\mathcal{L}f &= \frac{1}{2}\sum_{i,j=1}^{N} W_{ij}(f_i - f_j)^2 \\ &= \frac{1}{2}\sum_{i\in\mathcal{V}_1, j\in\mathcal{V}_1^c} W_{ij}\left(\sqrt{\frac{|\mathcal{V}_1^c|}{|\mathcal{V}_1|}} + \sqrt{\frac{|\mathcal{V}_1|}{|\mathcal{V}_1^c|}}\right)^2 + \frac{1}{2}\sum_{i\in\mathcal{V}_1^c, j\in\mathcal{V}_1} W_{ij}\left(-\sqrt{\frac{|\mathcal{V}_1^c|}{|\mathcal{V}_1|}} - \sqrt{\frac{|\mathcal{V}_1|}{|\mathcal{V}_1^c|}}\right)^2 \\ &= \left(\frac{|\mathcal{V}_1|}{|\mathcal{V}_1^c|} + \frac{|\mathcal{V}_1^c|}{|\mathcal{V}_1|} + 2\right)\sum_{i\in\mathcal{V}_1, j\in\mathcal{V}_1^c} W_{ij} = N \cdot \text{RatioCut}(\mathcal{V}_1, \mathcal{V}_1^c) \end{aligned}$$

Spectral Clustering (cont'd)

EXAMPLE: $k = 2$ (von Luxburg, 2007)

$$\min_{\mathcal{V}_1 \subset \mathcal{V}} \text{RatioCut}(\mathcal{V}_1, \mathcal{V}_1^c) \Leftrightarrow \min_{\mathcal{V}_1 \subset \mathcal{V}} f^{\mathrm{T}} \mathcal{L} f \text{ s.t. } f \perp \mathbf{1},\ \|f\|_2 = \sqrt{N}, \text{ and } f_i = \begin{cases} \sqrt{\frac{|\mathcal{V}_1^c|}{|\mathcal{V}_1|}} & , \text{ if } i \in \mathcal{V}_1 \\ -\sqrt{\frac{|\mathcal{V}_1|}{|\mathcal{V}_1^c|}} & , \text{ if } i \in \mathcal{V}_1^c \end{cases}$$

Spectral Clustering (cont'd)

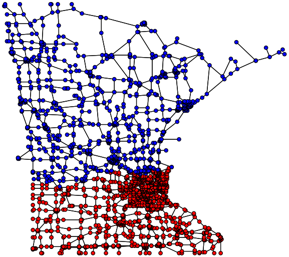
<u>EXAMPLE: $k = 2$</u> (von Luxburg, 2007)

- $\min_{\mathcal{V}_1 \subset \mathcal{V}} \text{RatioCut}(\mathcal{V}_1, \mathcal{V}_1^c) \Leftrightarrow \min_{\mathcal{V}_1 \subset \mathcal{V}} f^{\mathrm{T}} \mathcal{L} f$ s.t. $f \perp \mathbf{1}$, $\|f\|_2 = \sqrt{N}$, and $f_i = \begin{cases} \sqrt{\frac{|\mathcal{V}_1^c|}{|\mathcal{V}_1|}} & , \text{ if } i \in \mathcal{V}_1 \\ -\sqrt{\frac{|\mathcal{V}_1|}{|\mathcal{V}_1^c|}} & , \text{ if } i \in \mathcal{V}_1^c \end{cases}$
- NP hard, so we can relax the last condition: $\min_{\mathcal{V}_1 \subset \mathcal{V}} f^{\mathrm{T}} \mathcal{L} f$ s.t. $f \perp \mathbf{1}$ and $\|f\|_2 = \sqrt{N}$
- From the Courant-Fischer Theorem: $\chi_\ell = \operatorname*{argmin}_{x \perp \text{span}\left\{\chi_0, \ldots, \chi_{\ell-1}\right\},\ x \neq 0} \left\{ \frac{x^{\mathrm{T}} \mathcal{L} x}{x^{\mathrm{T}} x} \right\}$
- Thus, $f^* =$ Fiedler vector

Spectral Clustering (cont'd)

EXAMPLE: $k=2$ (von Luxburg, 2007)

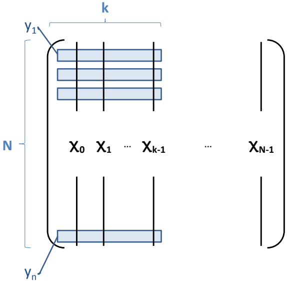
- $\min_{\mathcal{V}_1 \subset \mathcal{V}} \text{RatioCut}(\mathcal{V}_1, \mathcal{V}_1^c) \Leftrightarrow \min_{\mathcal{V}_1 \subset \mathcal{V}} f^{\mathrm{T}} \mathcal{L} f$ s.t. $f \perp \mathbf{1}$, $\|f\|_2 = \sqrt{N}$, and $f_i = \begin{cases} \sqrt{\frac{|\mathcal{V}_1^c|}{|\mathcal{V}_1|}} & \text{, if } i \in \mathcal{V}_1 \\ -\sqrt{\frac{|\mathcal{V}_1|}{|\mathcal{V}_1^c|}} & \text{, if } i \in \mathcal{V}_1^c \end{cases}$
- NP hard, so we can relax the last condition: $\min_{\mathcal{V}_1 \subset \mathcal{V}} f^{\mathrm{T}} \mathcal{L} f$ s.t. $f \perp \mathbf{1}$ and $\|f\|_2 = \sqrt{N}$
- From the Courant-Fischer Theorem: $\chi_\ell = \underset{x \perp \text{span}\{\chi_0,\ldots,\chi_{\ell-1}\},\; x \neq 0}{\text{argmin}} \left\{ \frac{x^{\mathrm{T}} \mathcal{L} x}{x^{\mathrm{T}} x} \right\}$
- Thus, $f^* =$ Fiedler vector
- Spectral clustering: $f_i^* \underset{i \in \mathcal{V}_1^c}{\overset{i \in \mathcal{V}_1}{\gtrless}} \tau$



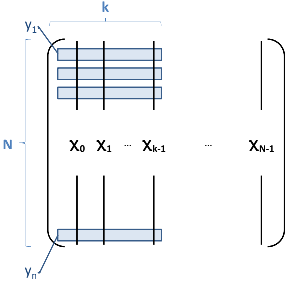
Spectral Clustering (cont'd)

GENERAL CASE: $k > 2$

- Form $\{y_i\}_{i=1,2,\ldots,N}$, where $y_i \in \mathbb{R}^k$
- Cluster y_i's with the k-means algorithm

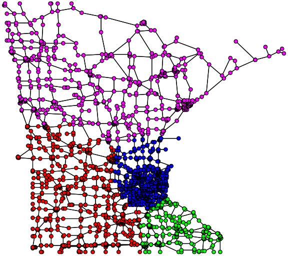


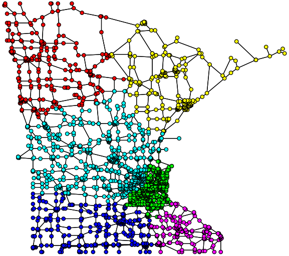
Spectral Clustering (cont'd)



GENERAL CASE: $k > 2$

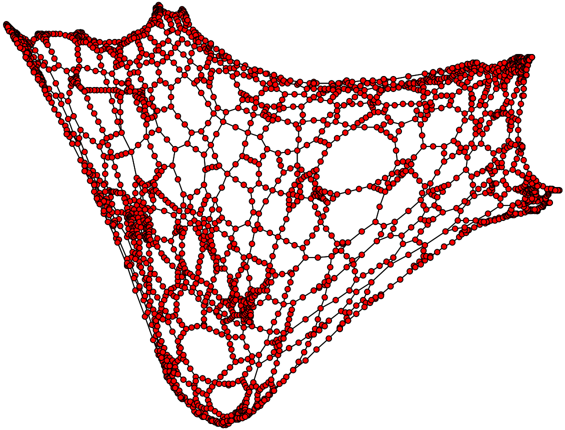
- Form $\{y_i\}_{i=1,2,\ldots,N}$, where $y_i \in \mathbb{R}^k$
- Cluster y_i's with the k-means algorithm

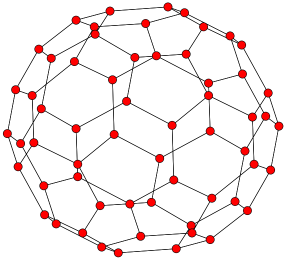




Graph Visualization

Use $\chi_1(i)$ and $\chi_2(i)$ as the x and y coordinates of the i^{th} vertex:





Source: Spielman, 2011

Outline

1 Introduction

2 Spectral Graph Theory Background

- Definitions
- Differential Operators on Graphs
- Graph Laplacian Eigenvectors
- Two Applications of Graph Laplacian Eigenvectors
- Graph Downsampling
- Filtering on Graphs

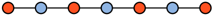
3 Wavelet Constructions on Graphs

4 Approximate Graph Multiplier Operators

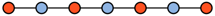
5 Distributed Signal Processing via the Chebyshev Approximation

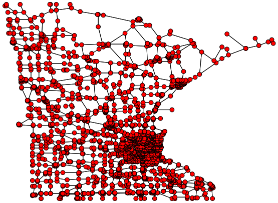
6 Open Issues and Challenges

Graph Downsampling



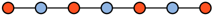
Graph Downsampling

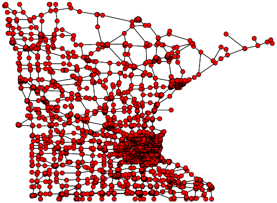




- Challenge: No clear notion of **every other vertex**

Graph Downsampling



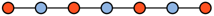


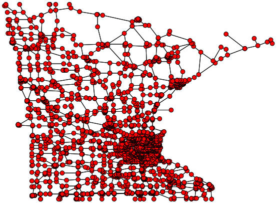
- Challenge: No clear notion of **every other vertex**

Wish List

- Removes approximately half of the vertices of the graph

Graph Downsampling



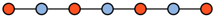


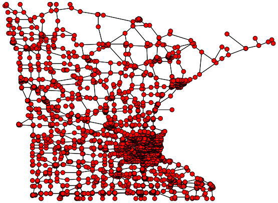
- Challenge: No clear notion of **every other vertex**

WISH LIST

- Removes approximately half of the vertices of the graph
- Eliminated vertices are not connected by edges of high weight
- Kept vertices are not connected by edges of high weight

Graph Downsampling





- Challenge: No clear notion of **every other vertex**

Wish List

- Removes approximately half of the vertices of the graph
- Eliminated vertices are not connected by edges of high weight
- Kept vertices are not connected by edges of high weight
- Can be implemented in a computationally efficient manner

Graph Downsampling
The Largest Eigenvector Method

- Downsample based on the polarity of the eigenvector associated with the largest eigenvalue of the graph Laplacian

- $\mathcal{V}_{keep} := \{i \in \mathcal{V} : \chi_{\max}(i) \geq 0\}\,,\ \mathcal{V}_{eliminate} := \{i \in \mathcal{V} : \chi_{\max}(i) < 0\}$

Graph Downsampling
The Largest Eigenvector Method

- Downsample based on the polarity of the eigenvector associated with the largest eigenvalue of the graph Laplacian
- $\mathcal{V}_{keep} := \{i \in \mathcal{V} : \chi_{\max}(i) \geq 0\}$, $\mathcal{V}_{eliminate} := \{i \in \mathcal{V} : \chi_{\max}(i) < 0\}$
- Variations: Keep negative, keep smallest or largest set, set threshold to something other than 0, use the largest eigenvector of the normalized Laplacian $\mathcal{L}^{norm}$

Graph Downsampling

The Largest Eigenvector Method

- Downsample based on the polarity of the eigenvector associated with the largest eigenvalue of the graph Laplacian
- $\mathcal{V}_{keep} := \{i \in \mathcal{V} : \chi_{\max}(i) \geq 0\}$, $\mathcal{V}_{eliminate} := \{i \in \mathcal{V} : \chi_{\max}(i) < 0\}$
- Variations: Keep negative, keep smallest or largest set, set threshold to something other than 0, use the largest eigenvector of the normalized Laplacian $\mathcal{L}^{norm}$
- Largest eigenvector efficiently computed with the power method:

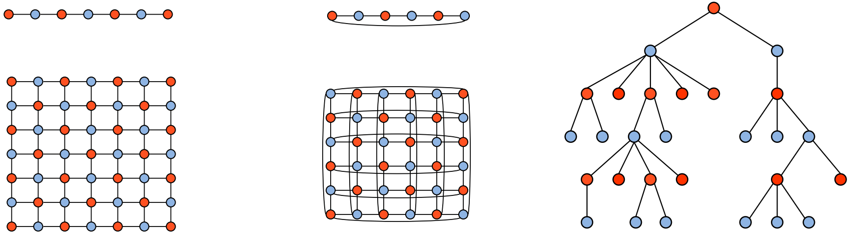
$$\mathbf{x}^{(k)} = \frac{\mathcal{L}\mathbf{x}^{(k-1)}}{\|\mathcal{L}\mathbf{x}^{(k-1)}\|_2}.$$

- If $\lambda_{\max} > \lambda_{N-1}$ and $\langle \mathbf{x}^{(0)}, \chi_{\max} \rangle \neq 0$, the sequence $\left\{\mathbf{x}^{(k)}\right\}_{k=0,1,\ldots}$ converges to $\chi_{\max}$

Graph Downsampling
The Largest Eigenvector Method – Examples

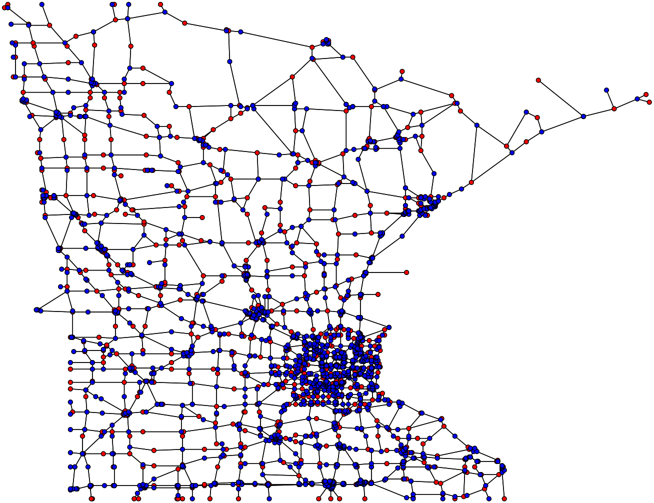
Theorem (Roth, 1989)

For a connected, bipartite graph $\mathcal{G} = \{\mathcal{V}_1 \cup \mathcal{V}_1^c, \mathcal{E}, \mathbf{W}\}$, the largest eigenvalues of $\mathcal{L}$ and $\mathcal{L}^{norm}$ are simple, and the polarities of the components of the eigenvectors $\chi_{\max}$ and $\chi_{\max}^{norm}$ split $\mathcal{V}$ into the bipartition $\mathcal{V}_1$ and $\mathcal{V}_1^c$.



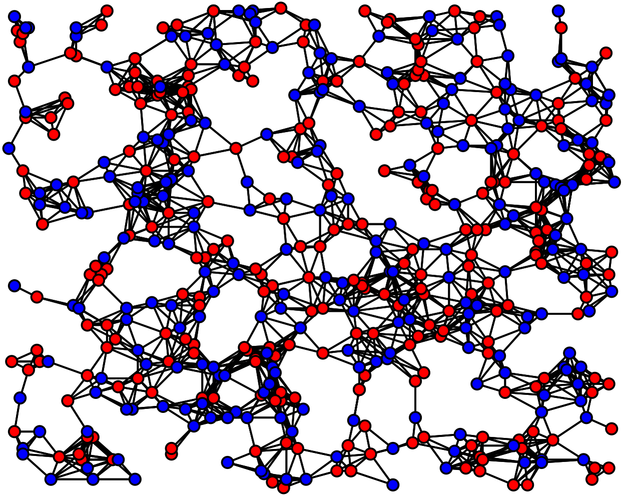
Graph Downsampling

The Largest Eigenvector Method – Examples



Graph Downsampling

The Largest Eigenvector Method – Examples



Graph Downsampling

Connections with Graph Coloring and Spectral Clustering

- A graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$ is *k-colorable* if there exists a partition of $\mathcal{V}$ into subsets $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_k$ such that if $i \sim j$, then i and j are in different subsets in the partition

- The chromatic number $\mathcal{C}$ of a graph $\mathcal{G}$ is the smallest k such that $\mathcal{G}$ is k-colorable

Graph Downsampling

Connections with Graph Coloring and Spectral Clustering

- A graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$ is *k-colorable* if there exists a partition of $\mathcal{V}$ into subsets $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_k$ such that if $i \sim j$, then i and j are in different subsets in the partition

- The chromatic number $\mathcal{C}$ of a graph $\mathcal{G}$ is the smallest k such that $\mathcal{G}$ is k-colorable

- The chromatic number is equal to 2 if and only if the graph is bipartite

Graph Downsampling

Connections with Graph Coloring and Spectral Clustering

- A graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$ is *k-colorable* if there exists a partition of $\mathcal{V}$ into subsets $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_k$ such that if $i \sim j$, then i and j are in different subsets in the partition

- The chromatic number $\mathcal{C}$ of a graph $\mathcal{G}$ is the smallest k such that $\mathcal{G}$ is k-colorable

- The chromatic number is equal to 2 if and only if the graph is bipartite

- In graph downsampling, we are interested in finding an *approximate 2-coloring* with few edges connecting vertices in the same subsets

Graph Downsampling

Connections with Graph Coloring and Spectral Clustering

- A graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$ is *k-colorable* if there exists a partition of $\mathcal{V}$ into subsets $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_k$ such that if $i \sim j$, then i and j are in different subsets in the partition

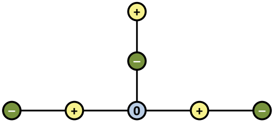
- The chromatic number $\mathcal{C}$ of a graph $\mathcal{G}$ is the smallest k such that $\mathcal{G}$ is k-colorable

- The chromatic number is equal to 2 if and only if the graph is bipartite

- In graph downsampling, we are interested in finding an *approximate 2-coloring* with few edges connecting vertices in the same subsets

- In some sense dual to the spectral clustering problem

Graph Downsampling
Connections with Nodal Domains

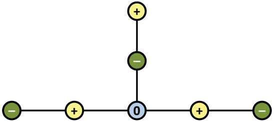


Source: Bıyıkoğlu *et al.*, 2007

- A *nodal domain* of a function f on $\mathcal{G}$ is a maximally connected subgraph of $\mathcal{G}$ such that the sign of f is the same on all vertices of the subgraph
- A positive (negative) *strong nodal domain* has $f(i) > 0$ ($f(i) < 0$) for all i in the subgraph
- A positive (negative) *weak nodal domain* has $f(i) \geq 0$ ($f(i) \leq 0$) for all i in the subgraph

Graph Downsampling

Connections with Nodal Domains



Source: Bıyıkoğlu *et al.*, 2007

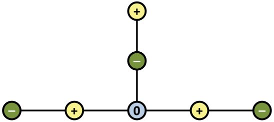
- A *nodal domain* of a function f on $\mathcal{G}$ is a maximally connected subgraph of $\mathcal{G}$ such that the sign of f is the same on all vertices of the subgraph

- A positive (negative) *strong nodal domain* has $f(i) > 0$ ($f(i) < 0$) for all i in the subgraph

- A positive (negative) *weak nodal domain* has $f(i) \geq 0$ ($f(i) \leq 0$) for all i in the subgraph

- $\#$ weak nodal domains of f on $\mathcal{G} \leq \#$ strong nodal domains of f on $\mathcal{G}$

Graph Downsampling
Connections with Nodal Domains



Source: Bıyıkoğlu *et al.*, 2007

- A *nodal domain* of a function f on $\mathcal{G}$ is a maximally connected subgraph of $\mathcal{G}$ such that the sign of f is the same on all vertices of the subgraph
- A positive (negative) *strong nodal domain* has $f(i) > 0$ ($f(i) < 0$) for all i in the subgraph
- A positive (negative) *weak nodal domain* has $f(i) \geq 0$ ($f(i) \leq 0$) for all i in the subgraph
- $\#$ weak nodal domains of f on $\mathcal{G} \leq \#$ strong nodal domains of f on $\mathcal{G}$
- Graph downsampling is closely related to the problem of maximizing the number of nodal domains

Graph Downsampling

Connections with Nodal Domains (cont'd)

General Bounds

- For any f on $\mathcal{G}$, $\#$ strong and weak nodal domains $\leq N - \mathcal{C} + 2$
- If $\mathcal{C} = 2$ ($\mathcal{G}$ is bipartite), $\exists f$ s.t. $\#$ strong and weak nodal domains of f is N

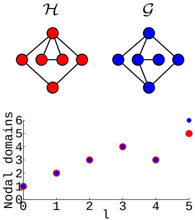
Graph Downsampling
Connections with Nodal Domains (cont'd)

General Bounds

- For any f on $\mathcal{G}$, # strong and weak nodal domains $\leq N - \mathcal{C} + 2$
- If $\mathcal{C} = 2$ ($\mathcal{G}$ is bipartite), $\exists f$ s.t. # strong and weak nodal domains of f is N

Bounds on the Nodal Domains of Laplacian Eigenvectors (Bıyıkoğlu *et al.*, 2007)

- # weak nodal domains of $\chi_\ell \leq \ell + 1$
- # strong nodal domains of $\chi_\ell \leq \ell + s$, where s is multiplicity of λ_ℓ
- $\chi_{\max}$ has N strong and weak nodal domains if and only if $\mathcal{G}$ is bipartite
- $\ell + 1 - r \leq$ # strong and weak nodal domains of χ_ℓ, if λ_ℓ is simple and $\chi_\ell(i) \neq 0$, $\forall i \in \mathcal{V}$, where r is the number of edges that need to be removed from the graph in order to turn it into a tree (Berkolaiko, 2008)



Source: Oren, 2007

Important Note

The bounds on the number of nodal domains of the Laplacian eigenvectors are monotonic in ℓ, but the actual number of nodal domains is not always monotonic in ℓ

Filtering on Graphs

- Filtering: represent an input signal as a combination of other signals, and amplify or attenuate the contributions of some of the component signals

Filtering on Graphs

- Filtering: represent an input signal as a combination of other signals, and amplify or attenuate the contributions of some of the component signals

- In classical signal processing, the most common choice of basis is the complex exponentials, which results in frequency filtering

Filtering on Graphs

- Filtering: represent an input signal as a combination of other signals, and amplify or attenuate the contributions of some of the component signals

- In classical signal processing, the most common choice of basis is the complex exponentials, which results in frequency filtering

- Not difficult to extend this notion to signals on graphs via the eigenvectors of the graph Laplacian

Graph Fourier Transform

- Fourier transform: expansion of f in terms of the eigenfunctions of the Laplacian / graph Laplacian

Functions on the Real Line

FOURIER TRANSFORM

$$\hat{f}(\omega) = \langle e^{i\omega x}, f \rangle = \int_{\mathbb{R}} f(x) e^{-i\omega x} \, dx$$

INVERSE FOURIER TRANSFORM

$$f(x) = \frac{1}{2\pi} \int_{\mathbb{R}} \hat{f}(\omega) e^{i\omega x} \, d\omega$$

Functions on the Vertices of a Graph

GRAPH FOURIER TRANSFORM

$$\hat{f}(\ell) = \langle \chi_\ell, f \rangle = \sum_{n=1}^{N} f(n) \chi_\ell^*(n)$$

INVERSE GRAPH FOURIER TRANSFORM

$$f(n) = \sum_{\ell=0}^{N-1} \hat{f}(\ell) \chi_\ell(n)$$

Fourier Multiplier Operator (Filter)

$$f(x) \longrightarrow \boxed{\text{FT}} \longrightarrow \hat{f}(\omega) \longrightarrow \boxed{g} \longrightarrow g(\omega)\hat{f}(\omega) \longrightarrow \boxed{\text{IFT}} \longrightarrow \Phi f(x)$$

- Fourier multiplier (filter) reshapes functions' frequencies:

$$\widehat{\Phi f}(\omega) = g(\omega)\hat{f}(\omega), \text{ for every frequency } \omega$$

Fourier Multiplier Operator (Filter)

$$\begin{tikzcd} f(x) \arrow[r] & \boxed{\text{FT}} \arrow[r] & \hat{f}(\omega) \arrow[r] & \boxed{g} \arrow[r] & g(\omega)\hat{f}(\omega) \arrow[r] & \boxed{\text{IFT}} \arrow[r] & \Phi f(x) \end{tikzcd}$$

- Fourier multiplier (filter) reshapes functions' frequencies:

$$\widehat{\Phi f}(\omega) = g(\omega)\hat{f}(\omega), \text{ for every frequency } \omega$$

- We can extend this to any group with a Fourier transform, including weighted, undirected graphs:

$$\Phi f = \text{IFT}\Big(g(\omega)\text{FT}(f)(\omega)\Big)$$

Functions on the Real Line

$$\Phi f(x) = \frac{1}{2\pi} \int_{\mathbb{R}} g(\omega)\hat{f}(\omega)e^{i\omega x} \, d\omega$$

Functions on the Vertices of a Graph

$$\Phi f(n) = \sum_{\ell=0}^{N-1} g(\lambda_\ell)\hat{f}(\ell)\chi_\ell(n)$$

Generalized Graph Multiplier Operators

- Graph Fourier transform leads to natural notions of smoothness
- However, we can just as easily use different filtering bases (useful in practice)

Generalized Graph Multiplier Operators

- Graph Fourier transform leads to natural notions of smoothness
- However, we can just as easily use different filtering bases (useful in practice)

Definition

$\mathbf{\Psi}$ is a graph multiplier operator with respect to the real symmetric positive semi-definite matrix $\mathbf{P}$ if there exists a function $g : [0, \lambda_{\max}(\mathbf{P})] \to \mathbb{R}$ and a complete set $\{\boldsymbol{\chi}_\ell\}_{\ell=0,1,\ldots,N-1}$ of orthonormal eigenvectors of $\mathbf{P}$ such that

$$\mathbf{\Psi} = \sum_{\ell=0}^{N-1} g(\lambda_\ell)\boldsymbol{\chi}_\ell\boldsymbol{\chi}_\ell^*,$$

where $\{\lambda_\ell\}_{\ell=0,1,\ldots,N-1}$ are the eigenvalues of $\mathbf{P}$.

Generalized Graph Multiplier Operators

- Graph Fourier transform leads to natural notions of smoothness
- However, we can just as easily use different filtering bases (useful in practice)

Definition

$\mathbf{\Psi}$ is a graph multiplier operator with respect to the real symmetric positive semi-definite matrix $\mathbf{P}$ if there exists a function $g : [0, \lambda_{\text{max}}(\mathbf{P})] \rightarrow \mathbb{R}$ and a complete set $\{\boldsymbol{\chi}_\ell\}_{\ell=0,1,\ldots,N-1}$ of orthonormal eigenvectors of $\mathbf{P}$ such that

$$\mathbf{\Psi} = \sum_{\ell=0}^{N-1} g(\lambda_\ell)\boldsymbol{\chi}_\ell\boldsymbol{\chi}_\ell^*,$$

where $\{\lambda_\ell\}_{\ell=0,1,\ldots,N-1}$ are the eigenvalues of $\mathbf{P}$.

Proposition (Equivalent characterizations of graph multiplier operators)

The following are equivalent:

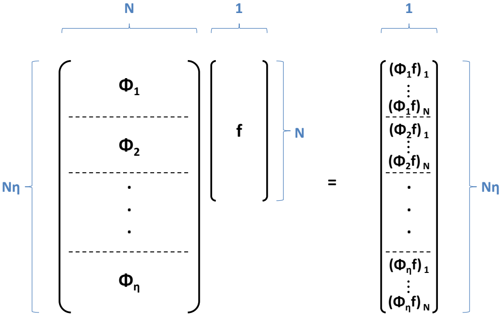
(a) $\mathbf{\Psi}$ *is a graph multiplier operator with respect to* $\mathbf{P}$.

(b) $\mathbf{\Psi}$ *and* $\mathbf{P}$ *are simultaneously diagonalizable by a unitary matrix; i.e., there exists a unitary matrix* $\mathbf{U}$ *such that* $\mathbf{U}^*\mathbf{\Psi}\mathbf{U}$ *and* $\mathbf{U}^*\mathbf{P}\mathbf{U}$ *are both diagonal matrices.*

(c) $\mathbf{\Psi}$ *and* $\mathbf{P}$ *commute; i.e.,* $\mathbf{\Psi}\mathbf{P} = \mathbf{P}\mathbf{\Psi}$.

Unions of Graph Multiplier Operators

- So far, just a single graph multiplier operator
- Can easily extend this to unions of graph multiplier operators:



Outline

1 Introduction

2 Spectral Graph Theory Background

3 Wavelet Constructions on Graphs

4 Approximate Graph Multiplier Operators

5 Distributed Signal Processing via the Chebyshev Approximation

6 Open Issues and Challenges

Transductive Learning

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

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 $|S| = l < n$

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)
- ...

Transductive 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

Transductive 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 unlabeled data be used ?

More general linear model with a dictionary of features ?

$$\|\mathbf{\Phi}_X\beta - \mathbf{y}\|_{2,S}^2 + \alpha\mathcal{S}(\beta)$$

dictionary depends on data points

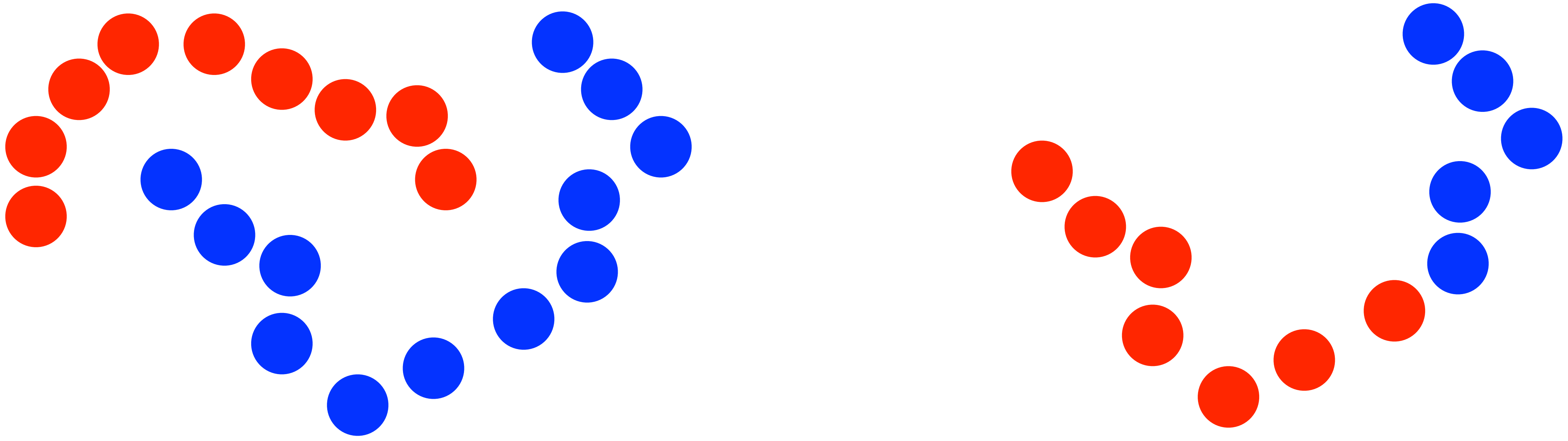
simplifies/stabilizes selected model

Learning on/with Graphs

How can unlabeled 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:
$$\begin{aligned}\Delta f &= \sum_{i,j \in X} \mathbf{W}_{ij}(f(x_i) - f(x_j))^2 \\ &= \mathbf{f}^t L \mathbf{f} \quad L = W - D\end{aligned}$$

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

Solution is smooth in graph "geometry"

Transduction & Representation

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 ?

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

Wavelet Ingredients

Wavelet transform based on two operations:

Dilation (or scaling) and Translation (or localization)

$$\psi_{s,a}(x) = \frac{1}{s}\psi\left(\frac{x-a}{s}\right)$$

Wavelet Ingredients

Wavelet transform based on two operations:

Dilation (or scaling) and Translation (or localization)

$$\psi_{s,a}(x) = \frac{1}{s}\psi\left(\frac{x-a}{s}\right)$$

$$(T^s f)(a) = \int \frac{1}{s}\psi^*\left(\frac{x-a}{s}\right) f(x)dx \qquad (T^s f)(a) = \langle \psi_{(s,a)}, f \rangle$$

Wavelet Ingredients

Wavelet transform based on two operations:

Dilation (or scaling) and Translation (or localization)

$$\psi_{s,a}(x) = \frac{1}{s}\psi\left(\frac{x-a}{s}\right)$$

$$(T^s f)(a) = \int \frac{1}{s}\psi^*\left(\frac{x-a}{s}\right) f(x)dx \qquad (T^s f)(a) = \langle \psi_{(s,a)}, f \rangle$$

Equivalently: $$(T^s \delta_a)(x) = \frac{1}{s}\psi^*\left(\frac{x-a}{s}\right)$$

$$(T^s f)(x) = \frac{1}{2\pi}\int e^{i\omega x}\hat{\psi}^*(s\omega)\hat{f}(\omega)d\omega$$

Graph Laplacian and Spectral Theory

$G = (V, E, w)$ weighted, undirected graph

Non-normalized Laplacian: $\mathcal{L} = D - A$ Real, symmetric

$$(\mathcal{L}f)(i) = \sum_{i \sim j} w_{i,j}(f(i) - f(j))$$

Why Laplacian ?

Graph Laplacian and Spectral Theory

$G = (V, E, w)$ weighted, undirected graph

Non-normalized Laplacian: $\mathcal{L} = D - A$ Real, symmetric

$$(\mathcal{L}f)(i) = \sum_{i \sim j} w_{i,j}(f(i) - f(j))$$

Why Laplacian ? $\mathbb{Z}^2$ with usual stencil

$$(\mathcal{L}f)_{i,j} = 4f_{i,j} - f_{i+1,j} - f_{i-1,j} - f_{i,j+1} - f_{i,j-1}$$

In general, graph laplacian from nicely sampled manifold converges to Laplace-Beltrami operator

Graph Laplacian and Spectral Theory

$G = (V, E, w)$ weighted, undirected graph

Non-normalized Laplacian: $\mathcal{L} = D - A$ Real, symmetric

$$(\mathcal{L}f)(i) = \sum_{i \sim j} w_{i,j}(f(i) - f(j))$$

Why Laplacian ? $\mathbb{Z}^2$ with usual stencil

$$(\mathcal{L}f)_{i,j} = 4f_{i,j} - f_{i+1,j} - f_{i-1,j} - f_{i,j+1} - f_{i,j-1}$$

In general, graph laplacian from nicely sampled manifold converges to Laplace-Beltrami operator

Remark:

$$\mathcal{L}^{norm} = D^{-1/2}\mathcal{L}D^{-1/2} = I - D^{-1/2}AD^{-1/2}$$

Graph Laplacian and Spectral Theory

$$\frac{d^2}{dx^2} \Rightarrow e^{i\omega x} \Rightarrow f(x) = \frac{1}{2\pi}\int \hat{f}(\omega)e^{i\omega x}d\omega$$

Graph Laplacian and Spectral Theory

$$\frac{d^2}{dx^2} \quad \Rightarrow \quad e^{i\omega x} \quad \Rightarrow \quad f(x) = \frac{1}{2\pi}\int \hat{f}(\omega)e^{i\omega x}d\omega$$

Eigen decomposition of Laplacian: $\mathcal{L}\phi_l = \lambda_l \phi_l$

Graph Laplacian and Spectral Theory

$$\frac{d^2}{dx^2} \quad \Rightarrow \quad e^{i\omega x} \quad \Rightarrow \quad f(x) = \frac{1}{2\pi}\int \hat{f}(\omega) e^{i\omega x} d\omega$$

Eigen decomposition of Laplacian: $\mathcal{L}\phi_l = \lambda_l \phi_l$

For simplicity assume connected graph and $0 = \lambda_0 < \lambda_1 \leq \lambda_2 ... \leq \lambda_{N-1}$

For any function on the vertex set (vector) we have:

$$\hat{f}(\ell) = \langle \phi_\ell, f \rangle = \sum_{i=1}^{N} \phi_\ell^*(i) f(i)$$ Graph Fourier Transform

$$f(i) = \sum_{\ell=0}^{N-1} \hat{f}(\ell) \phi_\ell(i)$$

Spectral Graph Wavelets

Remember good old Euclidean case:

$$(T^s f)(x) = \frac{1}{2\pi} \int e^{i\omega x} \hat{\psi}^*(s\omega) \hat{f}(\omega) d\omega$$

We will adopt this operator view

Spectral Graph Wavelets

Remember good old Euclidean case:

$$(T^s f)(x) = \frac{1}{2\pi} \int e^{i\omega x} \hat{\psi}^*(s\omega) \hat{f}(\omega) d\omega$$

We will adopt this operator view

Operator-valued function via continuous *Borel functional calculus*

$g : \mathbb{R}^+ \to \mathbb{R}^+$ $T_g = g(\mathcal{L})$ **Operator-valued function**

Action of operator is induced by its Fourier symbol

$$\widehat{T_g f}(\ell) = g(\lambda_\ell) \hat{f}(\ell) \qquad (T_g f)(i) = \sum_{\ell=0}^{N-1} g(\lambda_\ell) \hat{f}(\ell) \phi_\ell(i)$$

Spectral Graph Wavelets

$G{=}(E,V)$ a weighted undirected graph, with Laplacian $\mathcal{L} = D - A$

Spectral Graph Wavelets

$G=(E,V)$ a weighted undirected graph, with Laplacian $\mathcal{L} = D - A$

Dilation operates through operator: $T_g^t = g(t\mathcal{L})$

Spectral Graph Wavelets

$G{=}(E,V)$ a weighted undirected graph, with Laplacian $\mathcal{L} = D - A$

Dilation operates through operator: $T_g^t = g(t\mathcal{L})$

Translation (localization):

Define $\psi_{t,j} = T_g^t \delta_j$ response to a delta at vertex j

$$\psi_{t,j}(i) = \sum_{\ell=0}^{N-1} g(t\lambda_\ell)\phi_\ell^*(j)\phi_\ell(i) \qquad \mathcal{L}\phi_\ell(j) = \lambda_\ell\phi_\ell(j)$$

$$\psi_{t,a}(u) = \int_{\mathbb{R}} d\omega\, \hat{\psi}(t\omega)e^{-j\omega a}e^{j\omega u}$$

Spectral Graph Wavelets

$G{=}(E,V)$ a weighted undirected graph, with Laplacian $\mathcal{L} = D - A$

Dilation operates through operator: $T_g^t = g(t\mathcal{L})$

Translation (localization):

Define $\psi_{t,j} = T_g^t \delta_j$ response to a delta at vertex j

$$\psi_{t,j}(i) = \sum_{\ell=0}^{N-1} g(t\lambda_\ell)\phi_\ell^*(j)\phi_\ell(i) \qquad \mathcal{L}\phi_\ell(j) = \lambda_\ell\phi_\ell(j)$$

$$\psi_{t,a}(u) = \int_{\mathbb{R}} d\omega\, \hat{\psi}(t\omega) e^{-j\omega a} e^{j\omega u}$$

And so formally define the graph wavelet coefficients of f:

$$W_f(t,j) = \langle \psi_{t,j}, f \rangle \qquad W_f(t,j) = T_g^t f(j) = \sum_{\ell=0}^{N-1} g(t\lambda_\ell)\hat{f}(\ell)\phi_\ell(j)$$

Frames

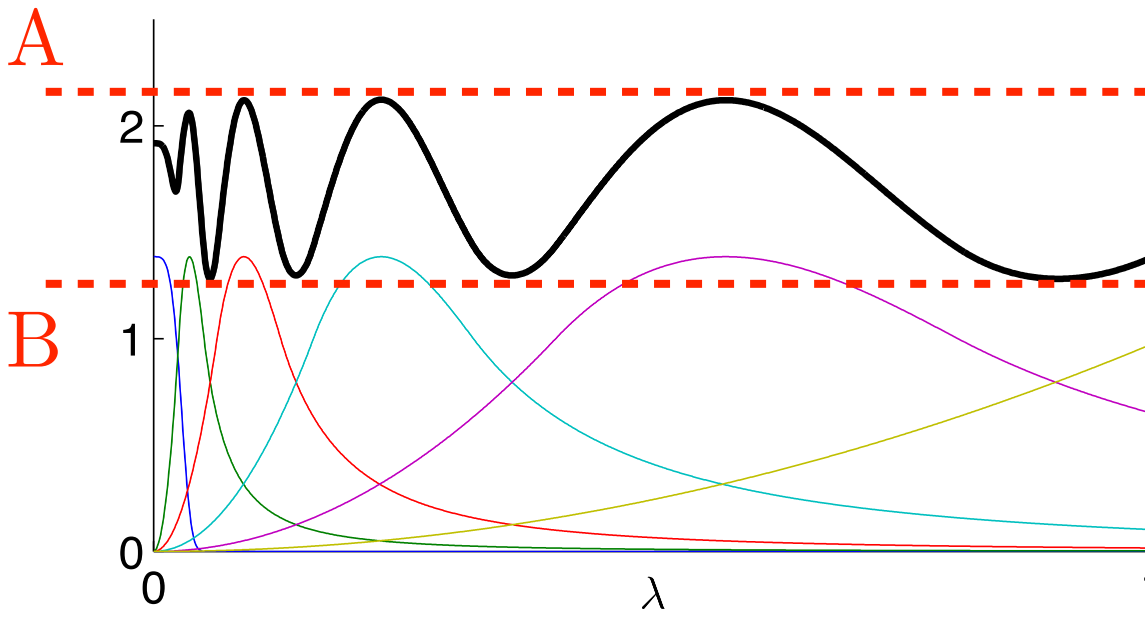
$\exists\, A, B > O,\ \exists h : \mathbb{R}_+ \to \mathbb{R}_+$ (i.e. scaling function)

$$0 < A \ \leqslant\ h^2(u)\ +\ \textstyle\sum_s g(t_s u)^2\ \leqslant\ B\ <\ \infty$$

scaling function

wavelets

$\phi_n = T_h \delta_n = h(\mathcal{L})\delta_n$

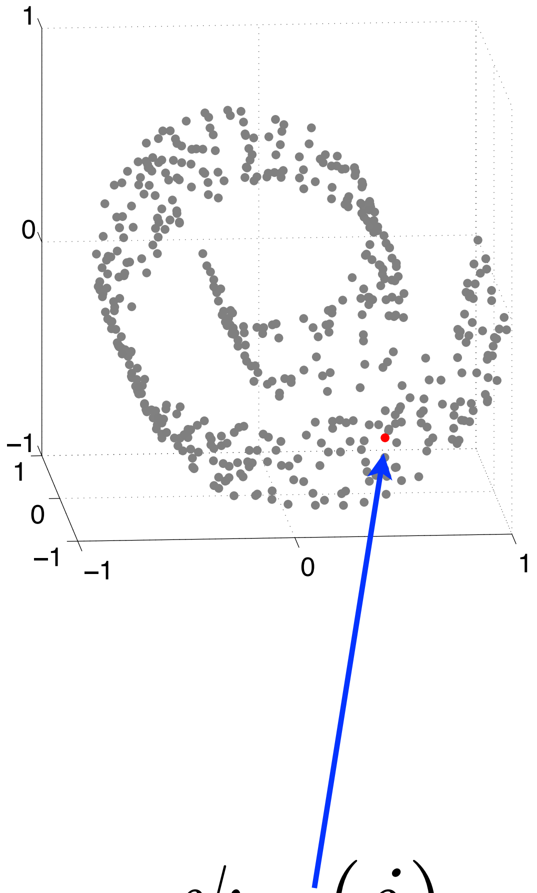


A simple way to get a tight frame:

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

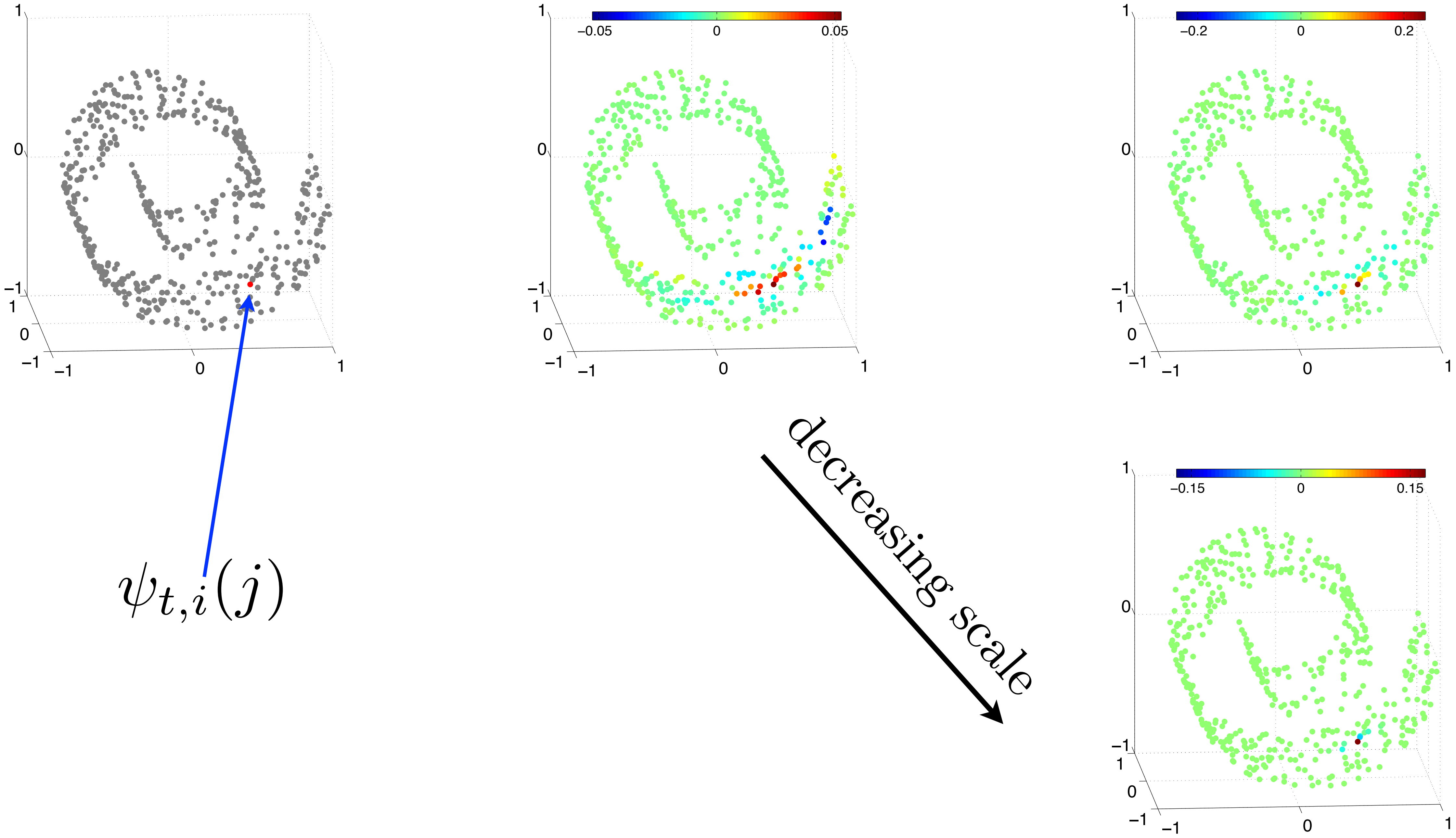
for any admissible kernel g

Scaling & Localization



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

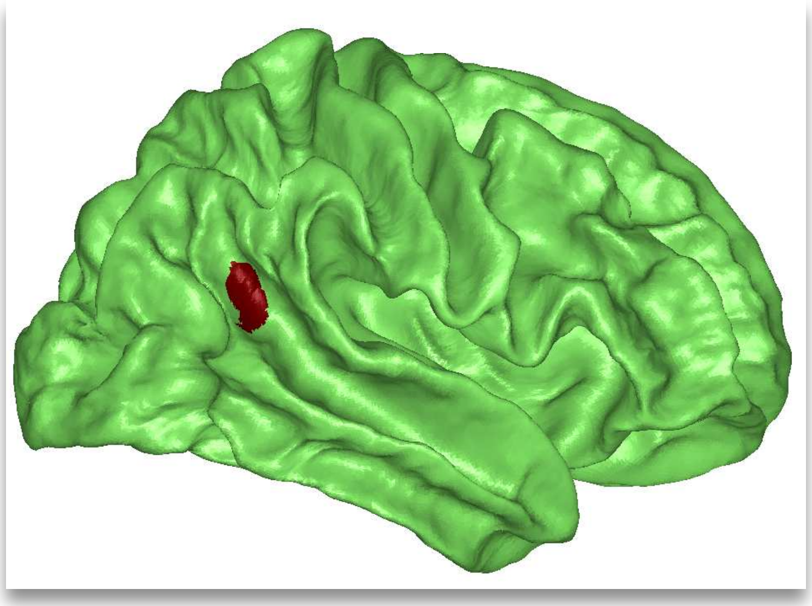
Scaling & Localization



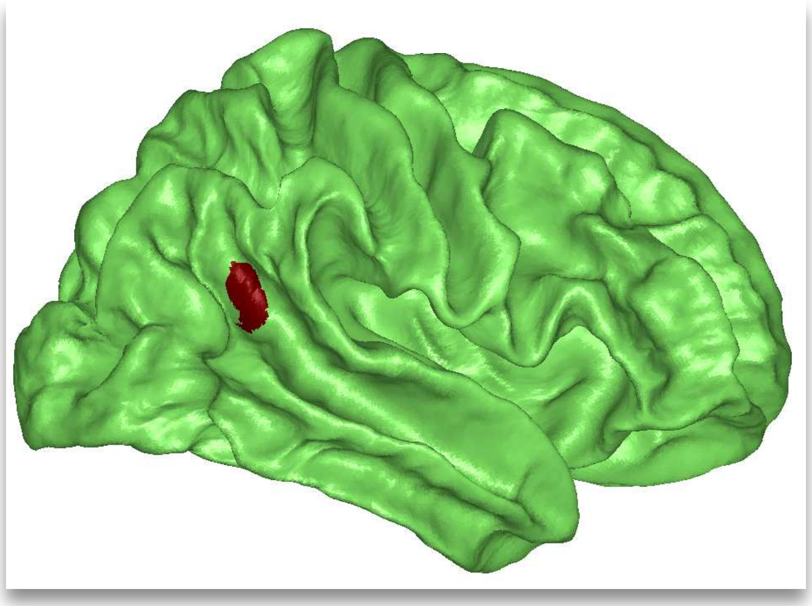
$\psi_{t,i}(j)$

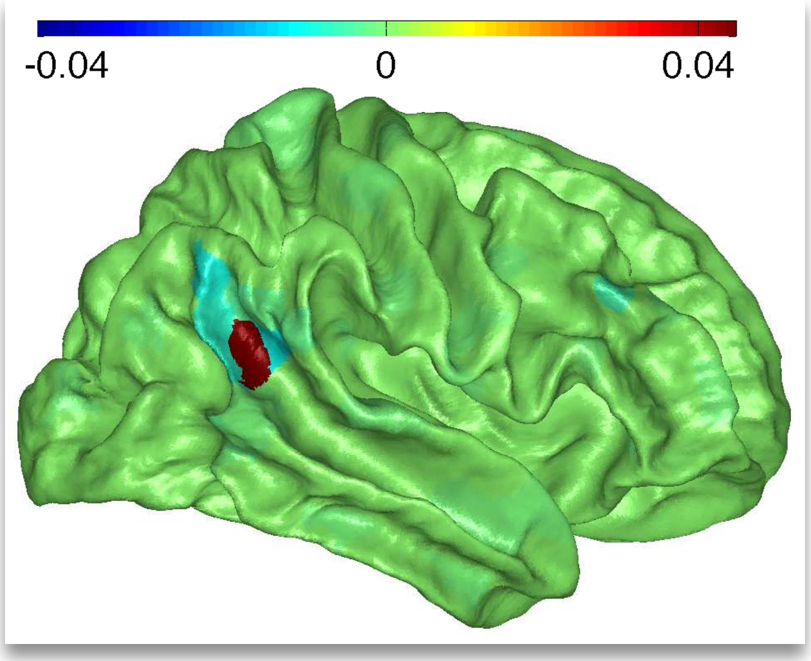
decreasing scale

Example

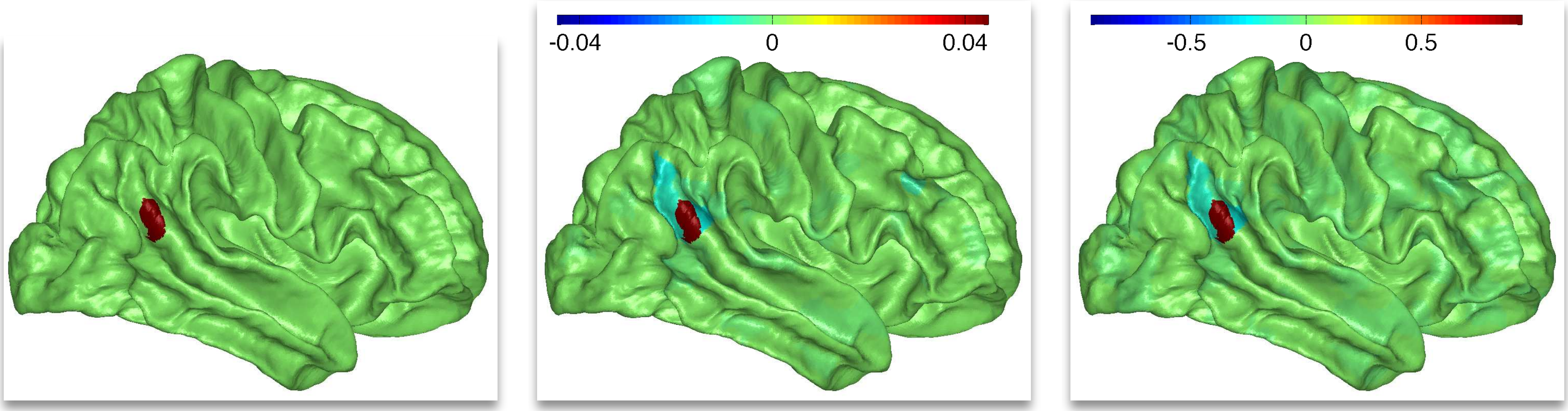


Example

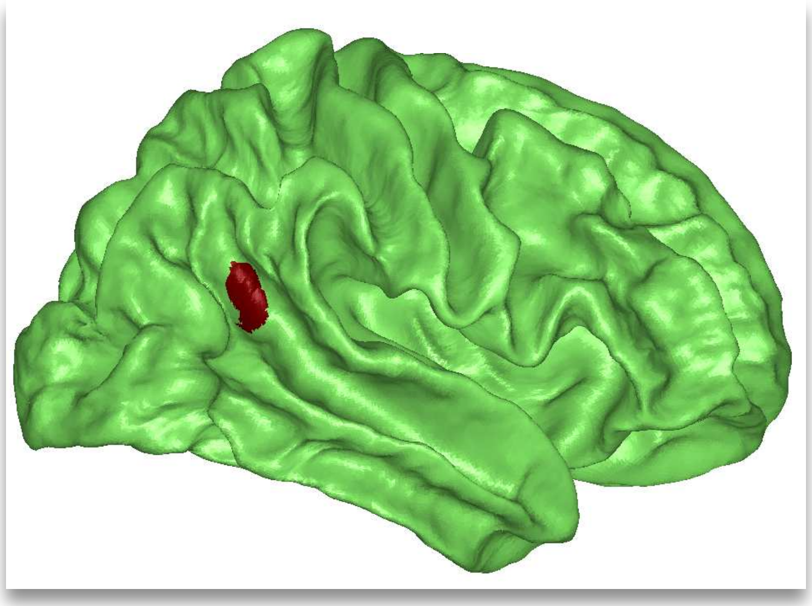


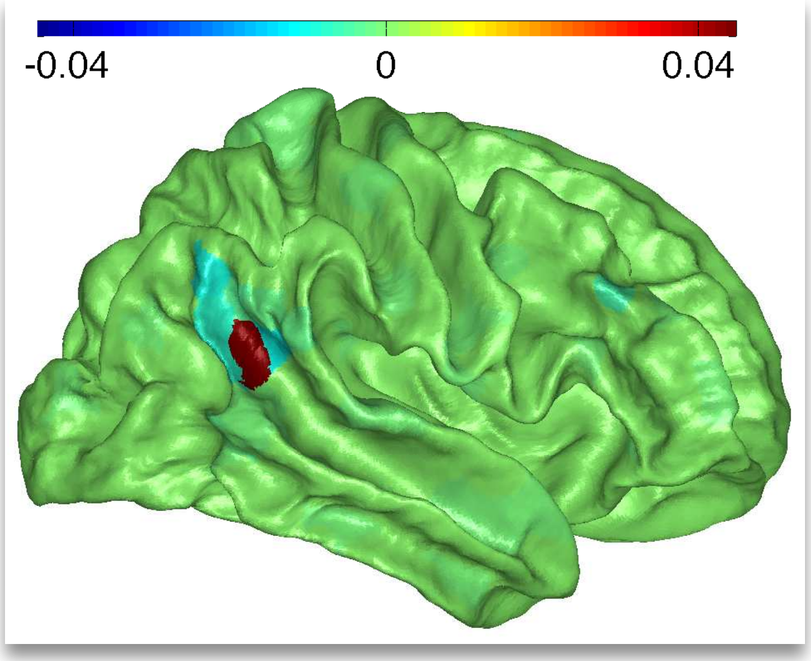


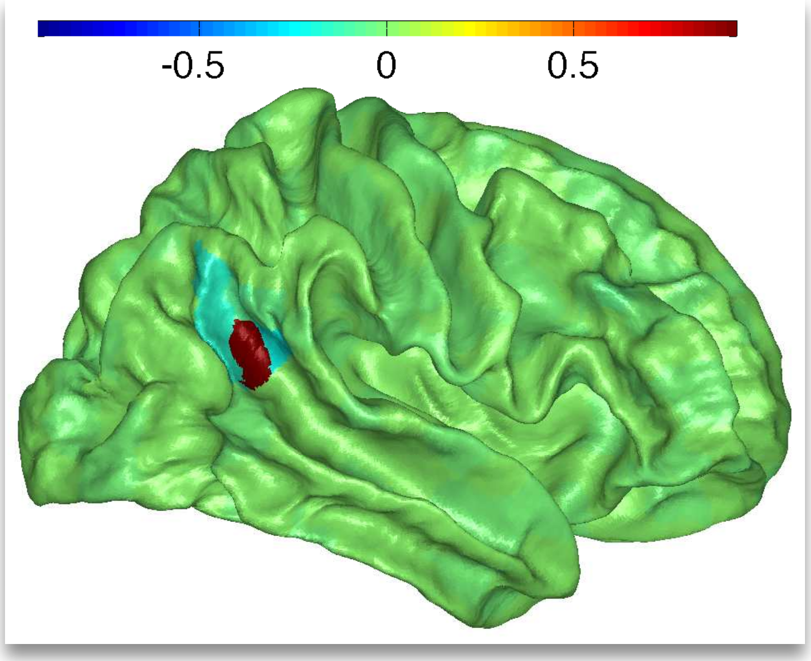
Example

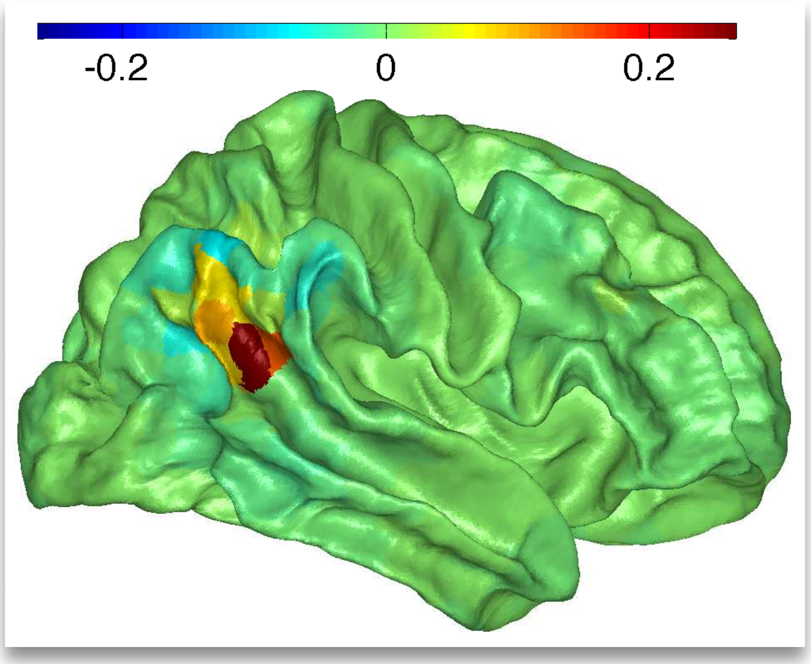


Example

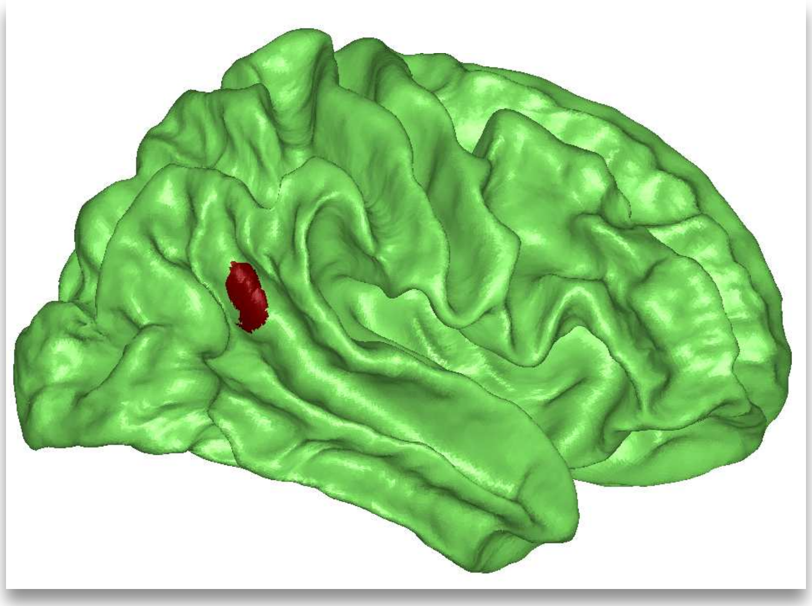


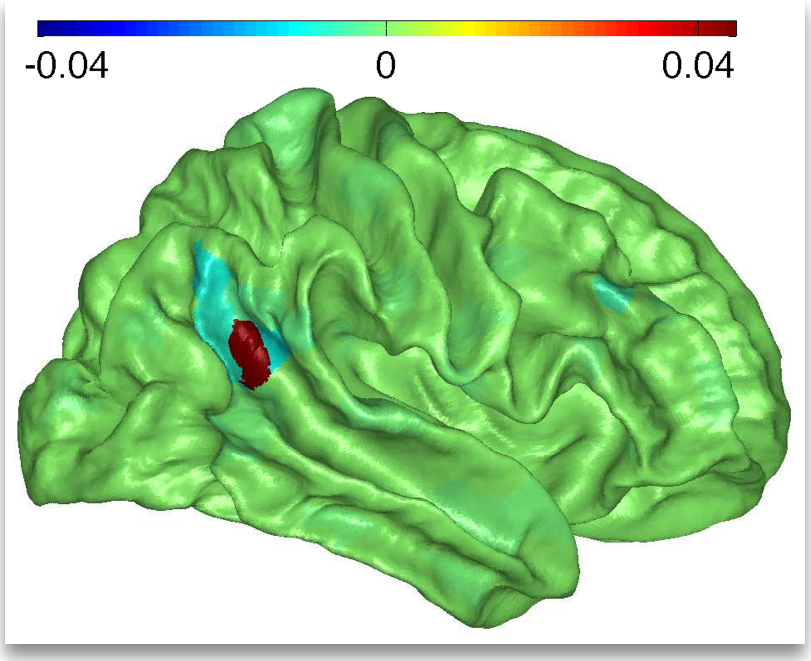


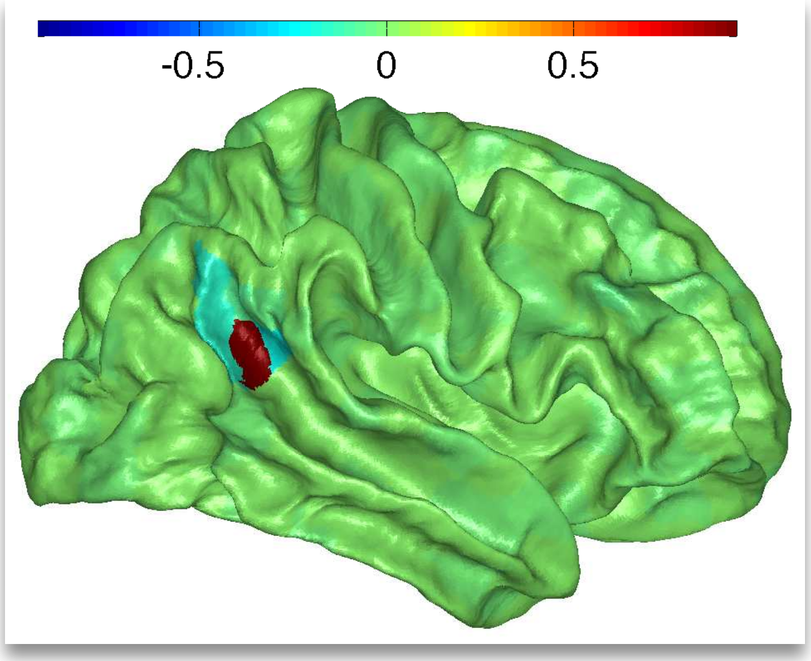


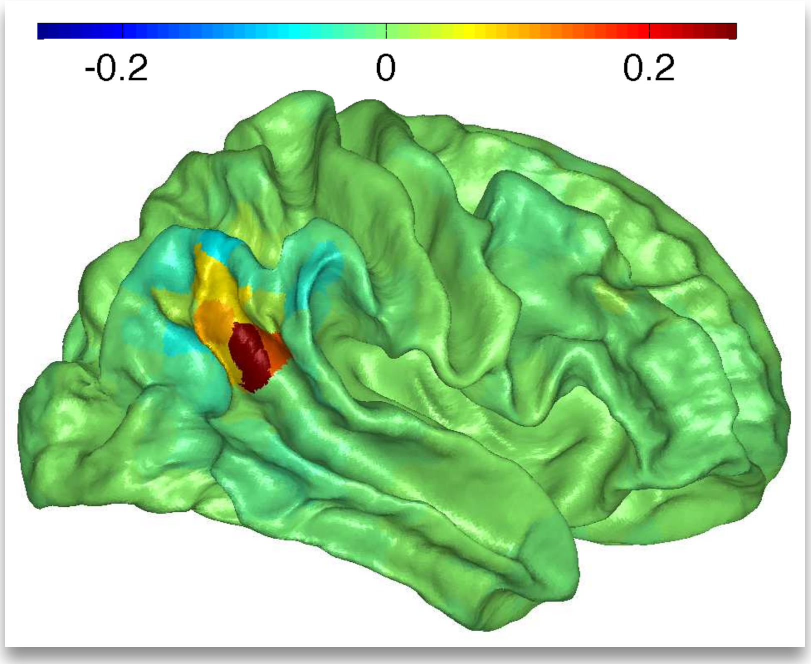


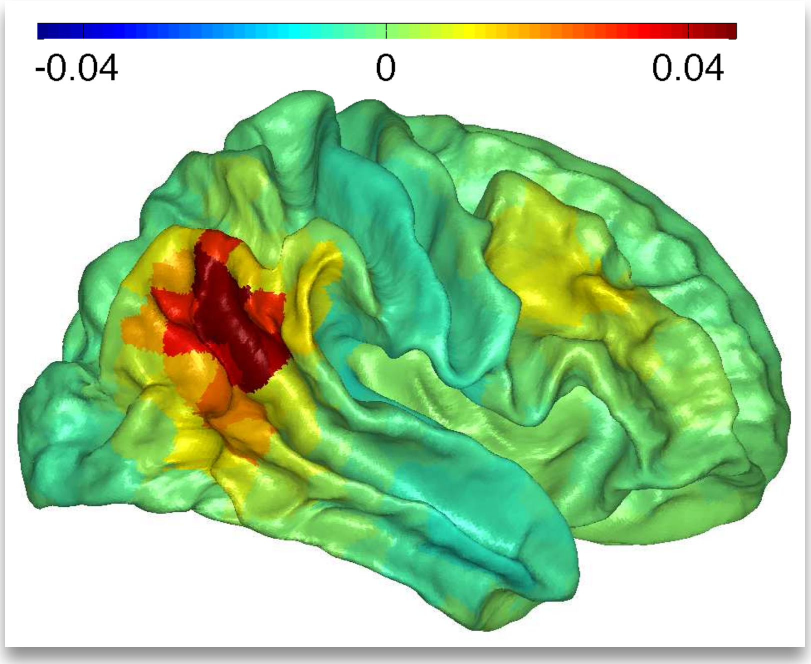
Example



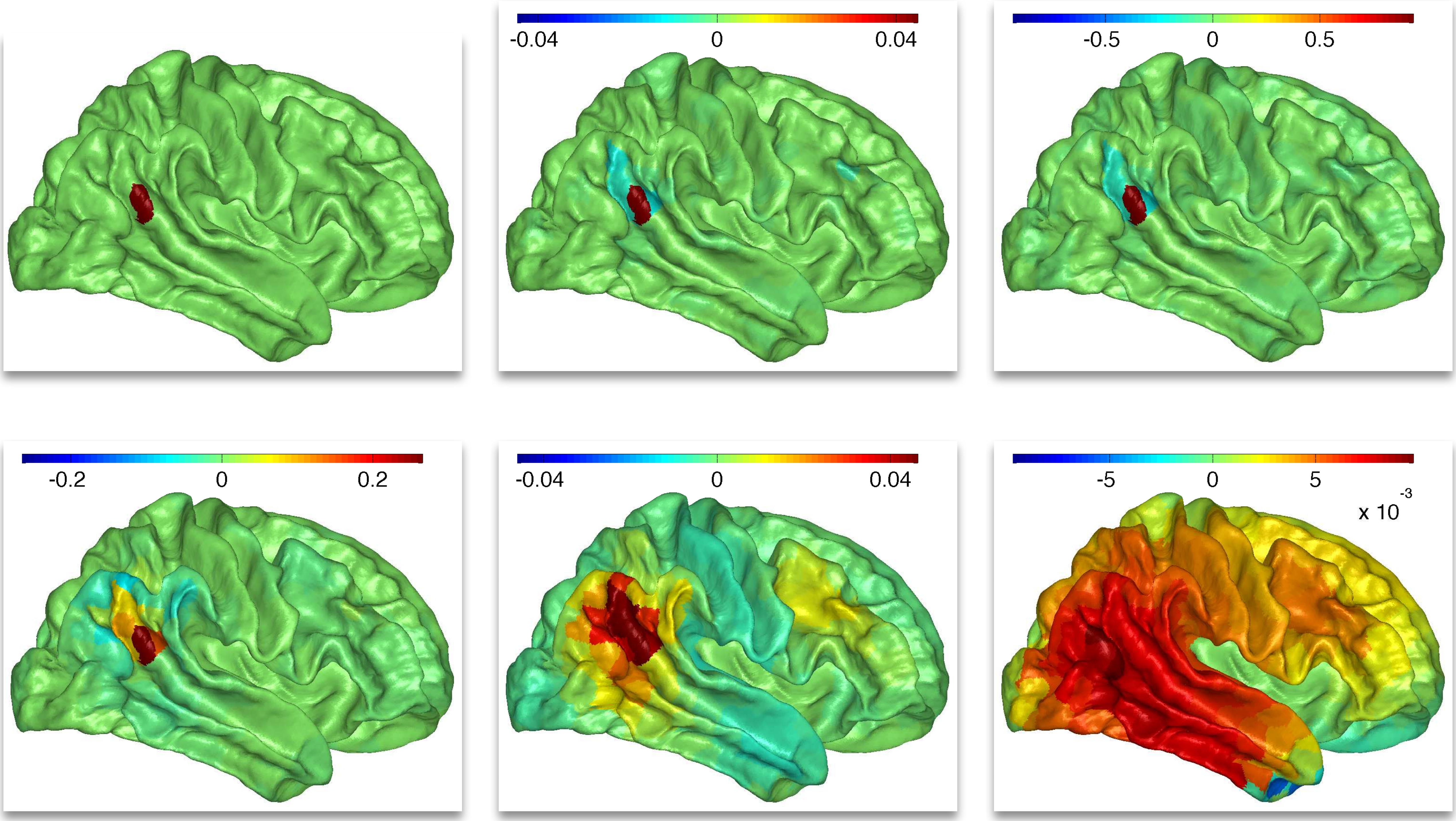


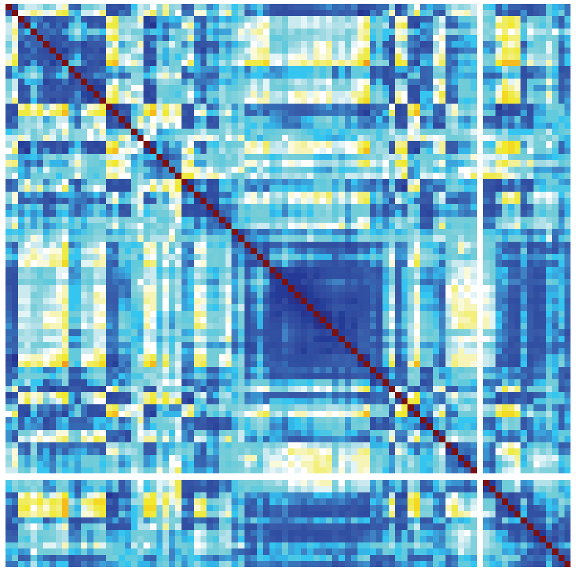




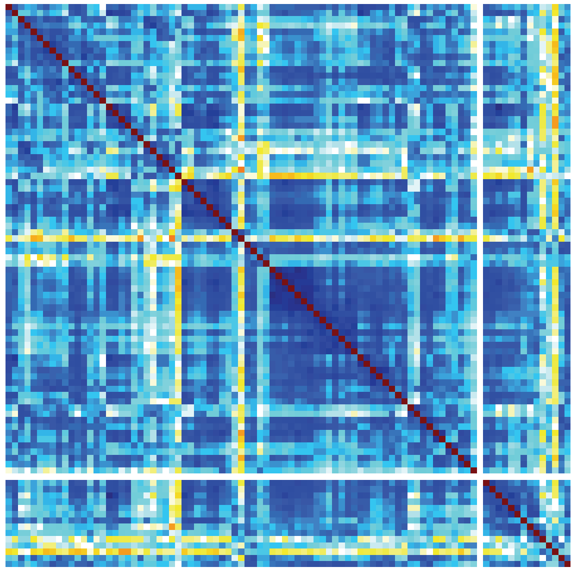


Example

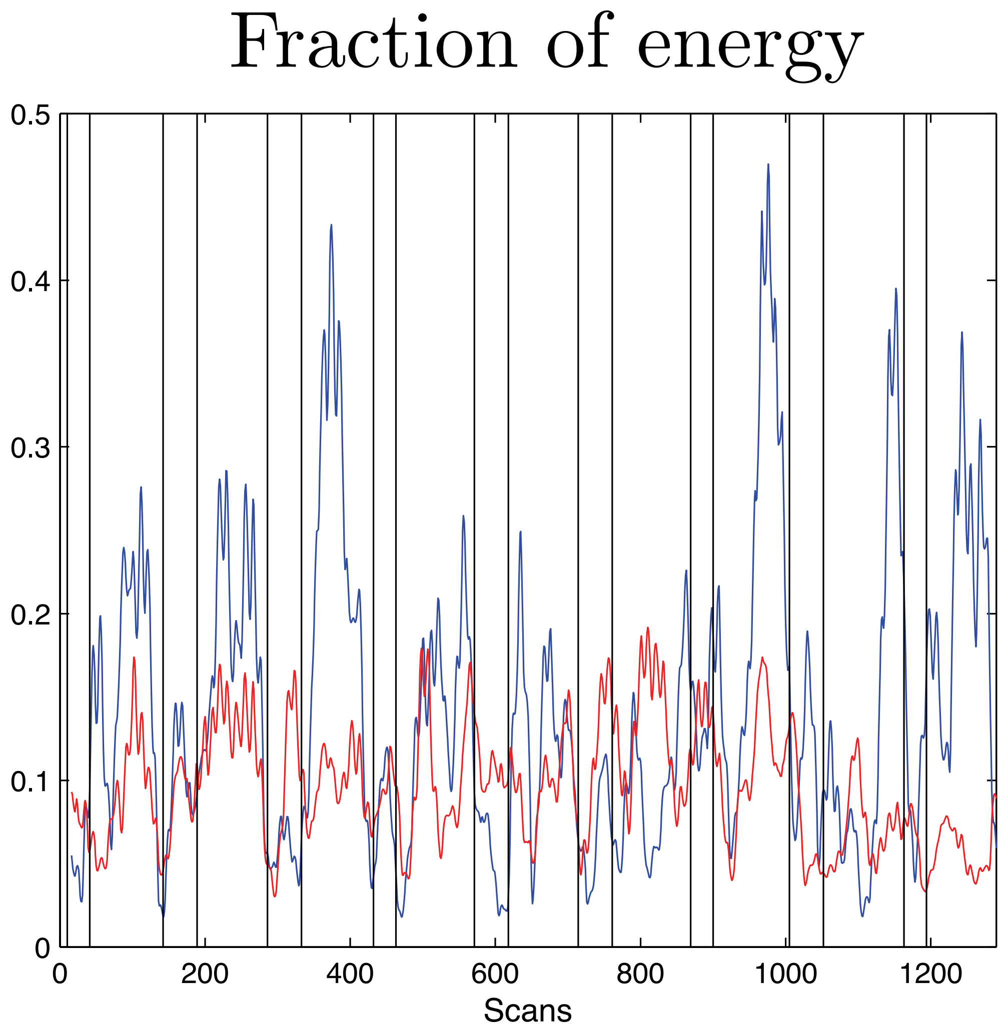




Rest



Movie



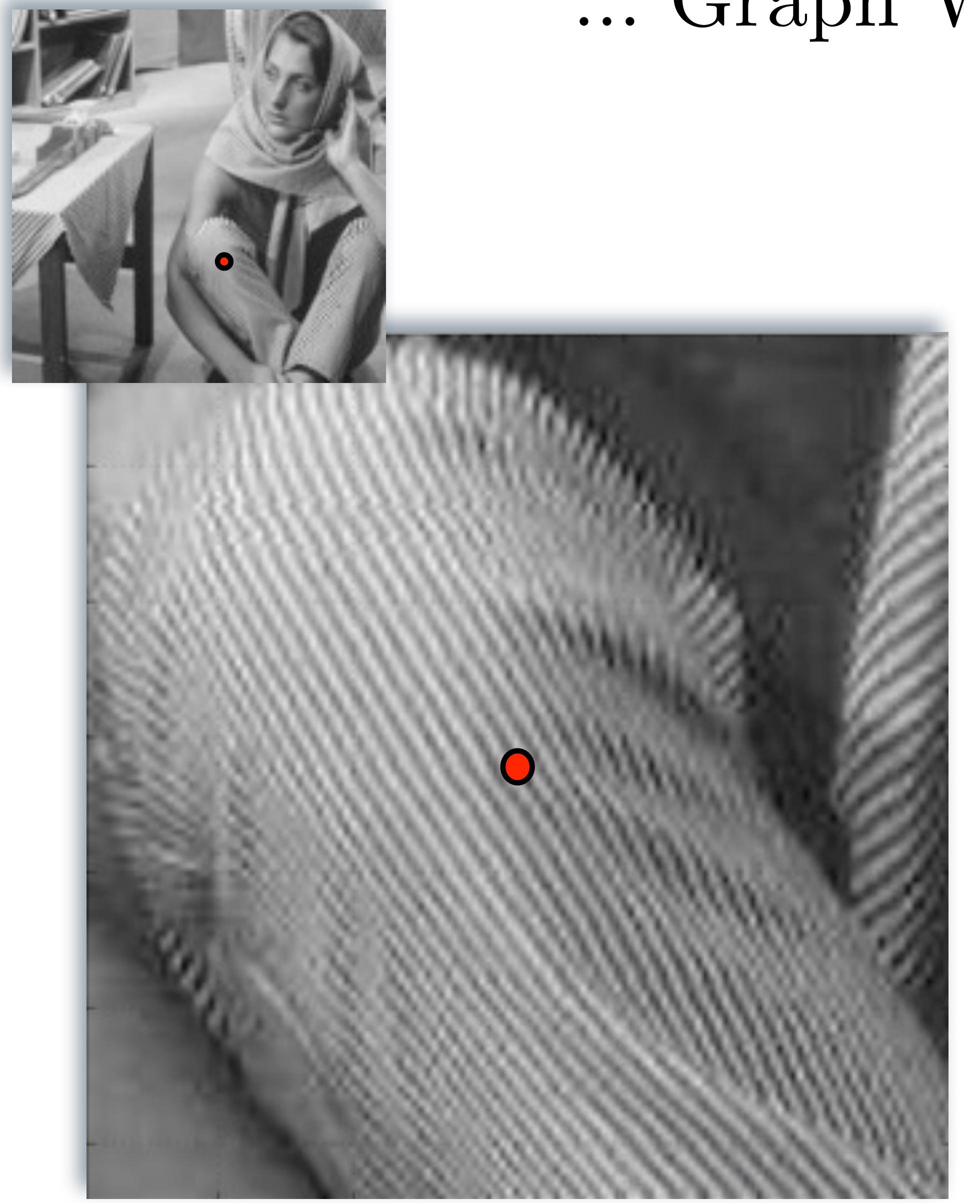
Fraction of energy

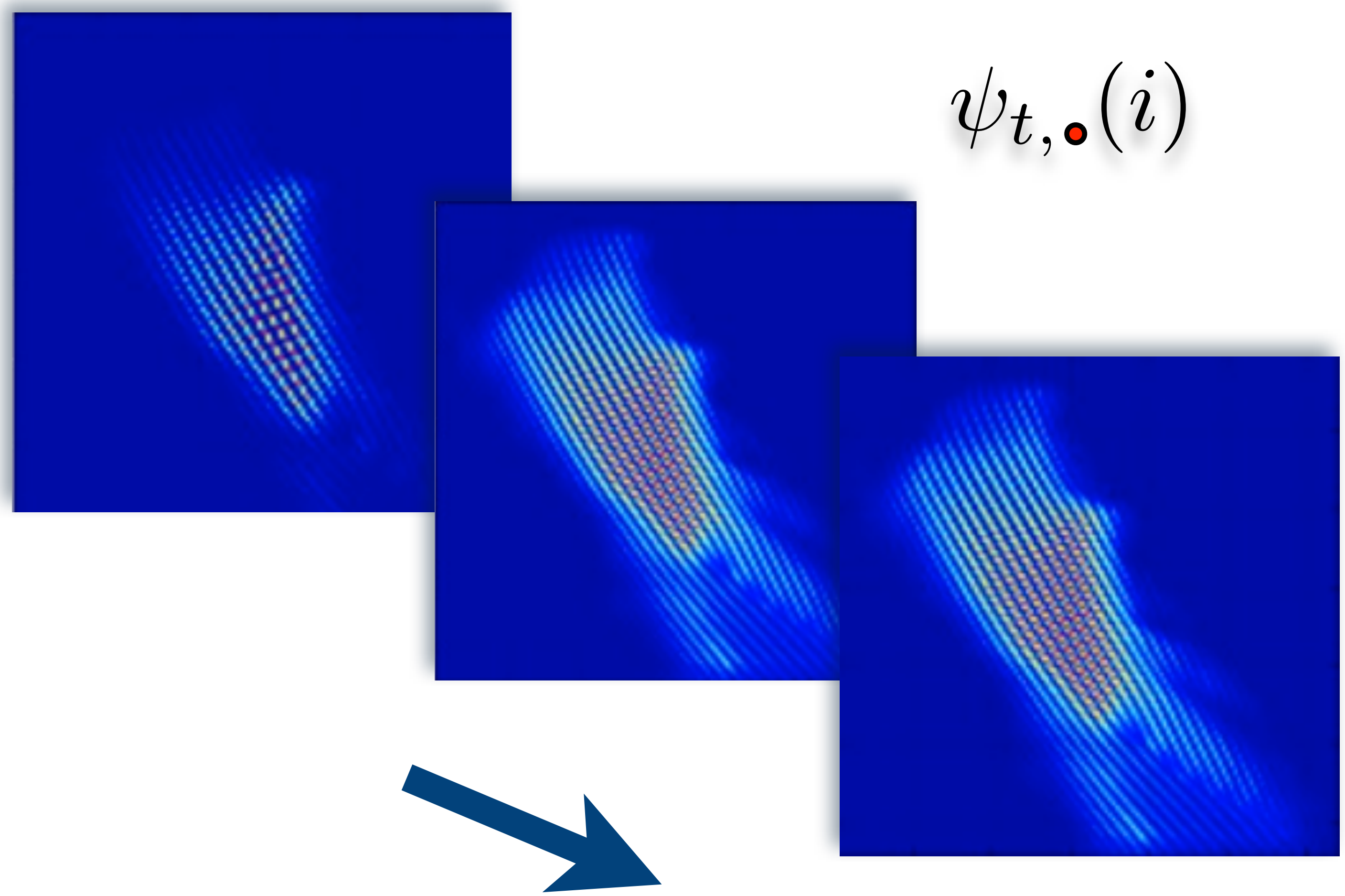
Leonardi & Van de Ville, 2011

Non-local Wavelet Frame

- Non-local Wavelets are ...

... Graph Wavelets on Non-Local Graph

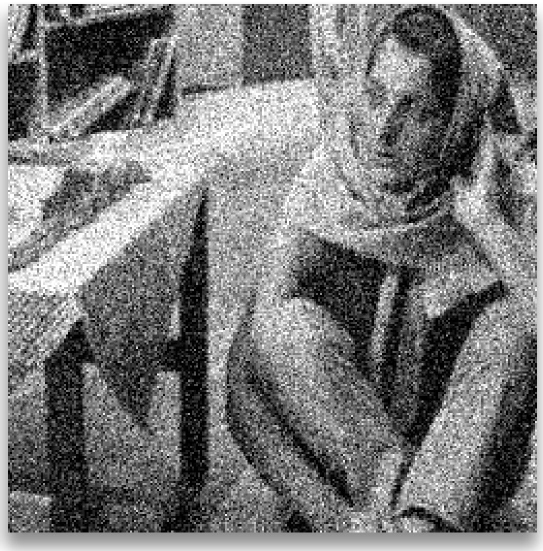




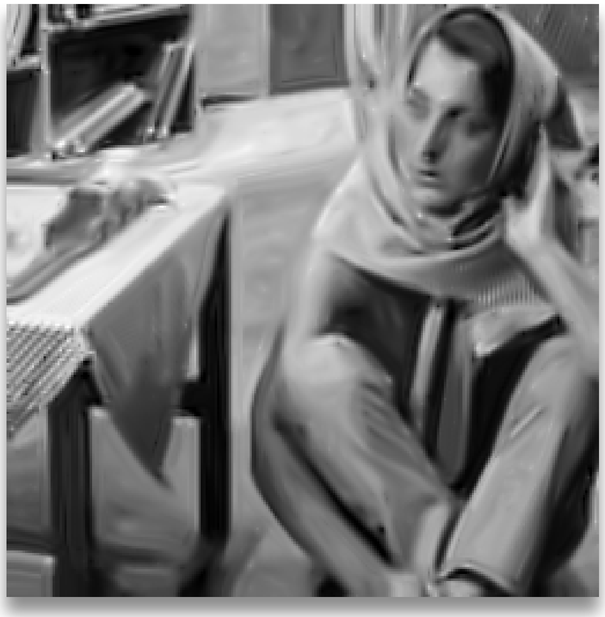
$\psi_{t,\bullet}(i)$

increasing scale

Interest: good *adaptive* sparsity basis



16.10dB



28.85dB

Sparsity and Smoothness on Graphs

Using a dictionary of graph wavelets, sparsity and smoothness on graphs are the same thing !

Sparsity and Smoothness on Graphs

Using a dictionary of graph wavelets, sparsity and smoothness on graphs are the same thing !

Idea: for a "Meyer kernel" on the spectrum of G

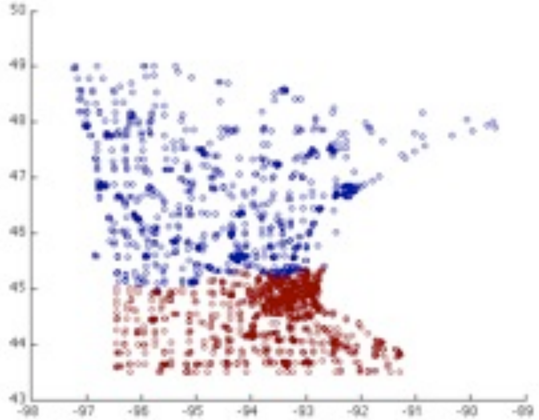
$$\sum_{i \in V} |\langle \psi_{2^{-j},i}, f \rangle|^2 = \sum_l |g(2^j \lambda_l)|^2 |\hat{f}(\lambda_l)|^2$$

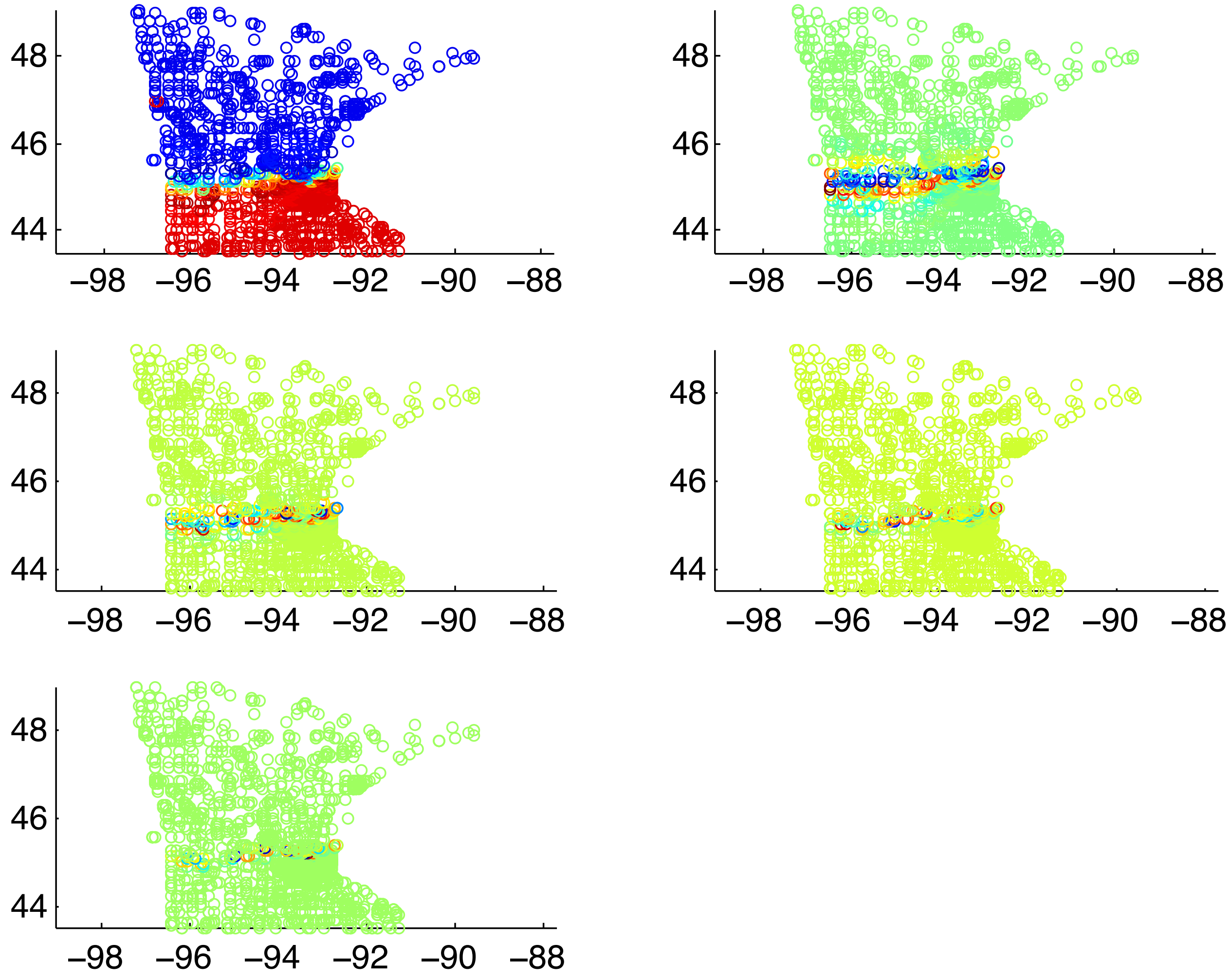
$$= \sum_{2^{-j-1}\lambda_{\max} \leq \lambda_l \leq 2^{-j}\lambda_{\max}} |\hat{f}(\lambda_l)|^2$$

$$A \sum_l \lambda_l^{2s} |\hat{f}(\lambda_l)|^2 \leq \sum_j 2^{-2sj} \sum_i |\langle \psi_{2^{-j},i}, f \rangle|^2 \leq B \sum_l \lambda_l^{2s} |\hat{f}(\lambda_l)|^2$$

$$\|f\|^2_{G,2s} = \sum_l \lambda_l^{2s} |\hat{f}(\lambda_l)|^2$$ discrete Sobolev semi-norm on G

Sparsity and Smoothness on Graphs




scaling functions coeffs

Sparsity and Transduction

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

Since sparsity = smoothness on graph, why not simple LASSO ?

$$\arg\min_{\beta} \|\mathbf{y} - \mathbf{M}\mathbf{\Phi}_X \beta\|_2^2 + \alpha \|\beta\|_1$$

Sparsity and Transduction

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

Since sparsity = smoothness on graph, why not simple LASSO ?

$$\arg\min_{\beta} \|\mathbf{y} - \mathbf{M}\mathbf{\Phi}_X \beta\|_2^2 + \alpha \|\beta\|_1$$

Bad Idea:

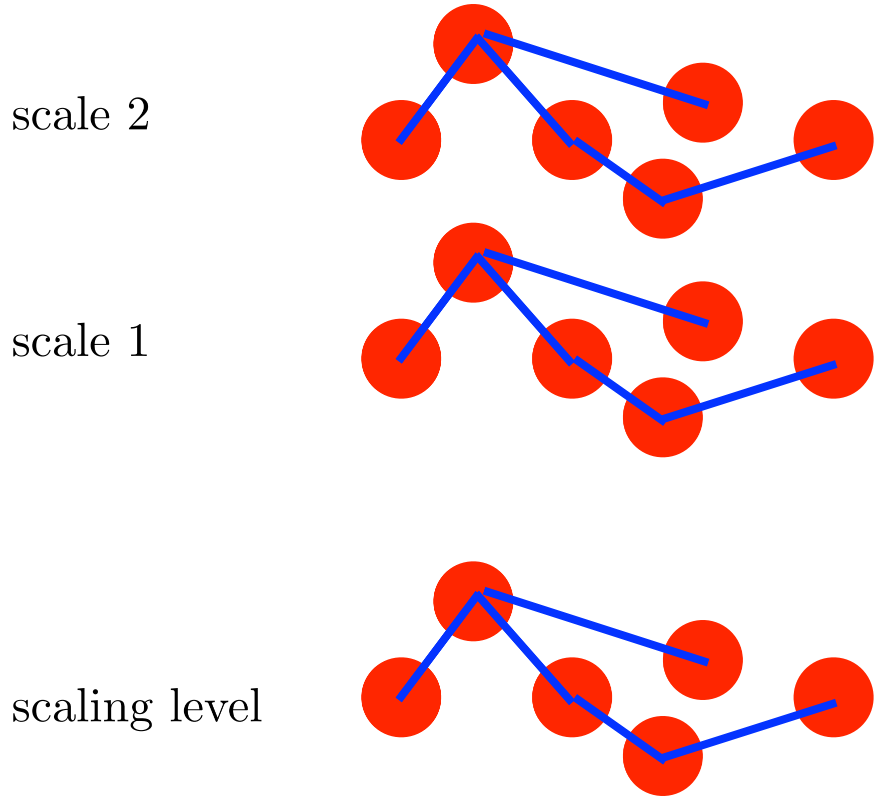
We *know* there are strongly correlated coefficients (LASSO will kill some of them)

There is no information to determine masked wavelets

Group Sparsity - take I

Scaling functions not sparse are optimized separately

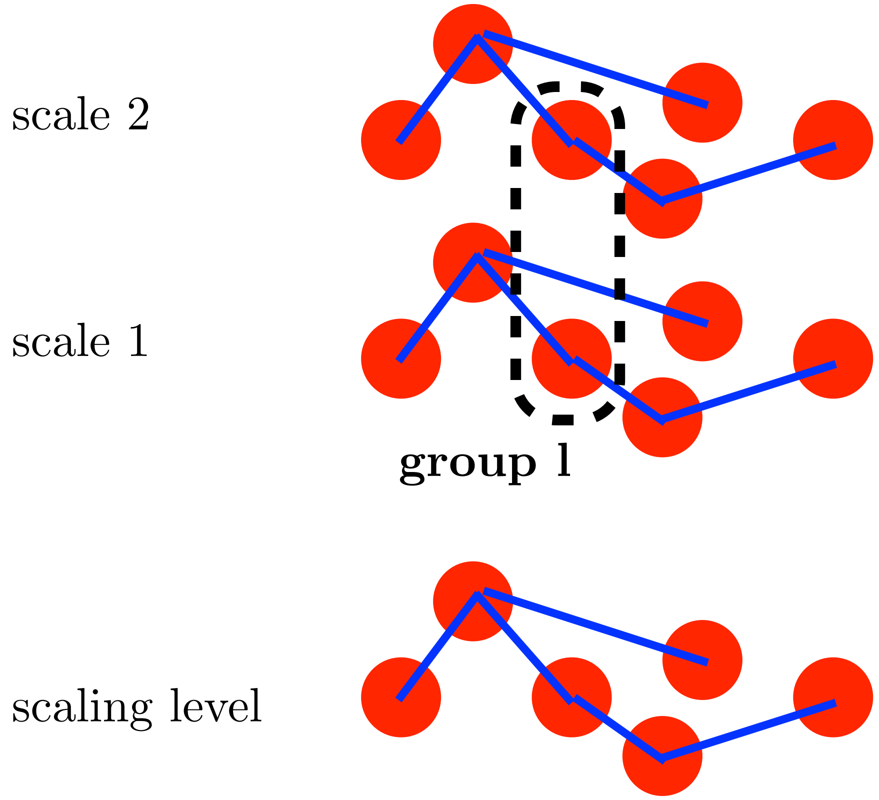
Group potentially correlated variables (scales)



Group Sparsity - take I

Scaling functions not sparse are optimized separately

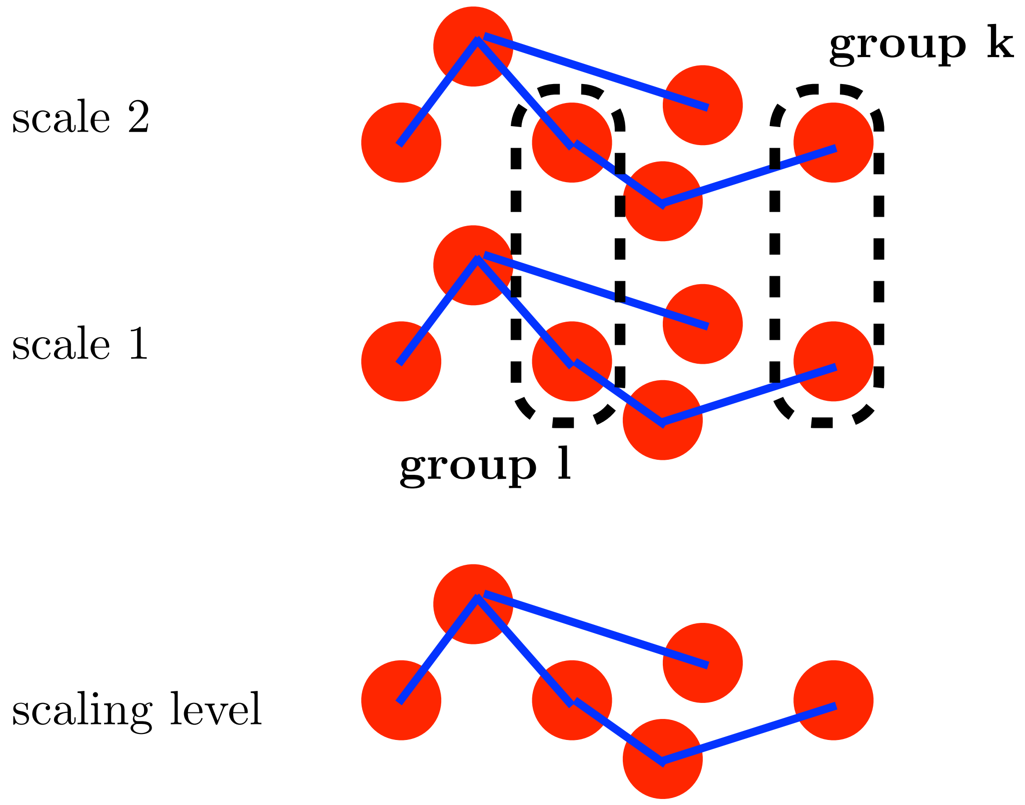
Group potentially correlated variables (scales)



Group Sparsity - take I

Scaling functions not sparse are optimized separately

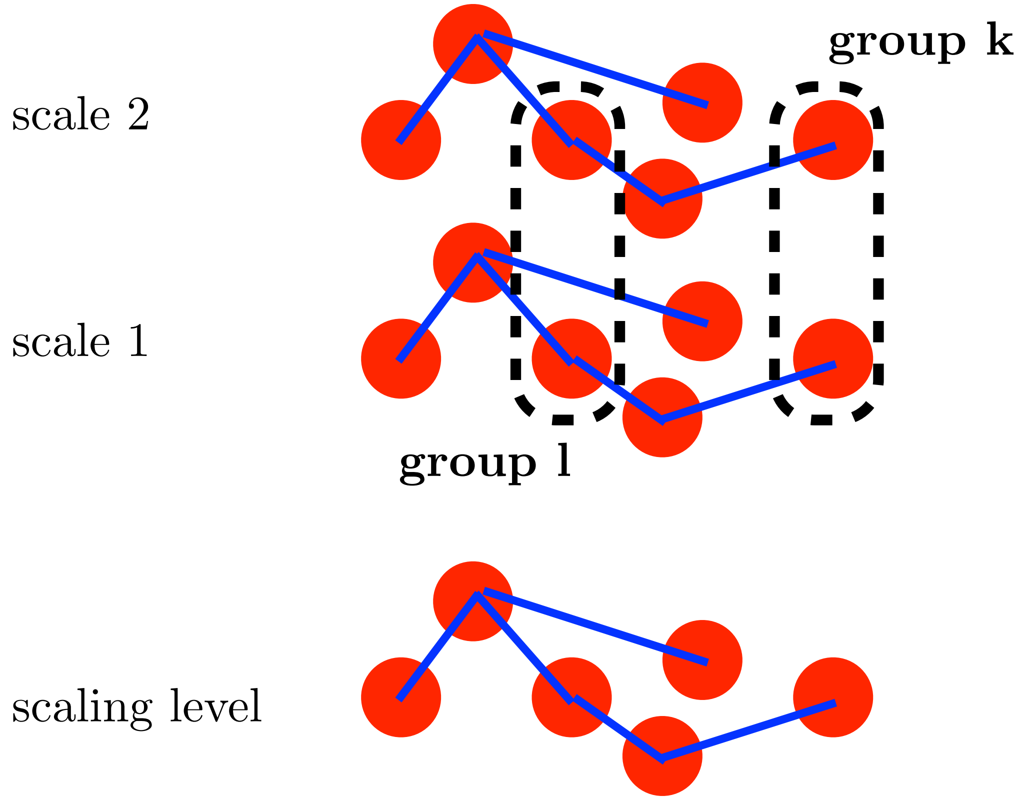
Group potentially correlated variables (scales)



Group Sparsity - take I

Scaling functions not sparse are optimized separately

Group potentially correlated variables (scales)



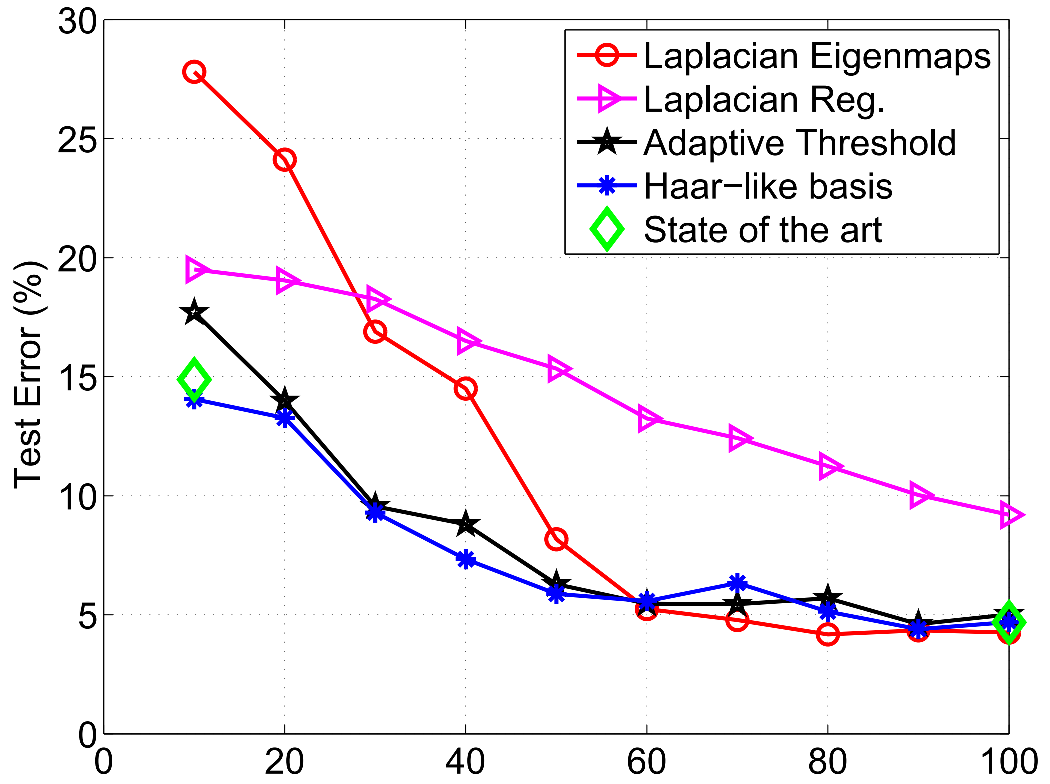
Few groups should be active = local smoothness

Inside group, all coefficients can be active

Formulate with mixed-norms $\|\beta\|_{p,q}$

Simple model, no overlap, optimized like LASSO

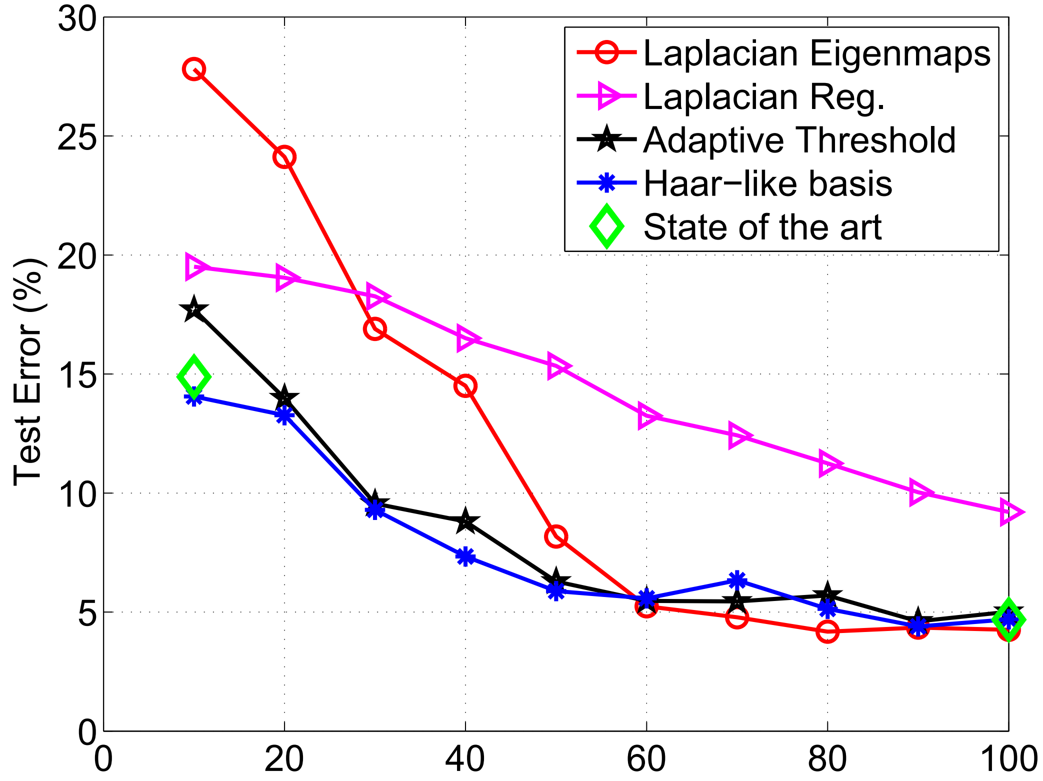
Preliminary Results



2-class USPS

Simulation results from Gavish et al, ICML 2010

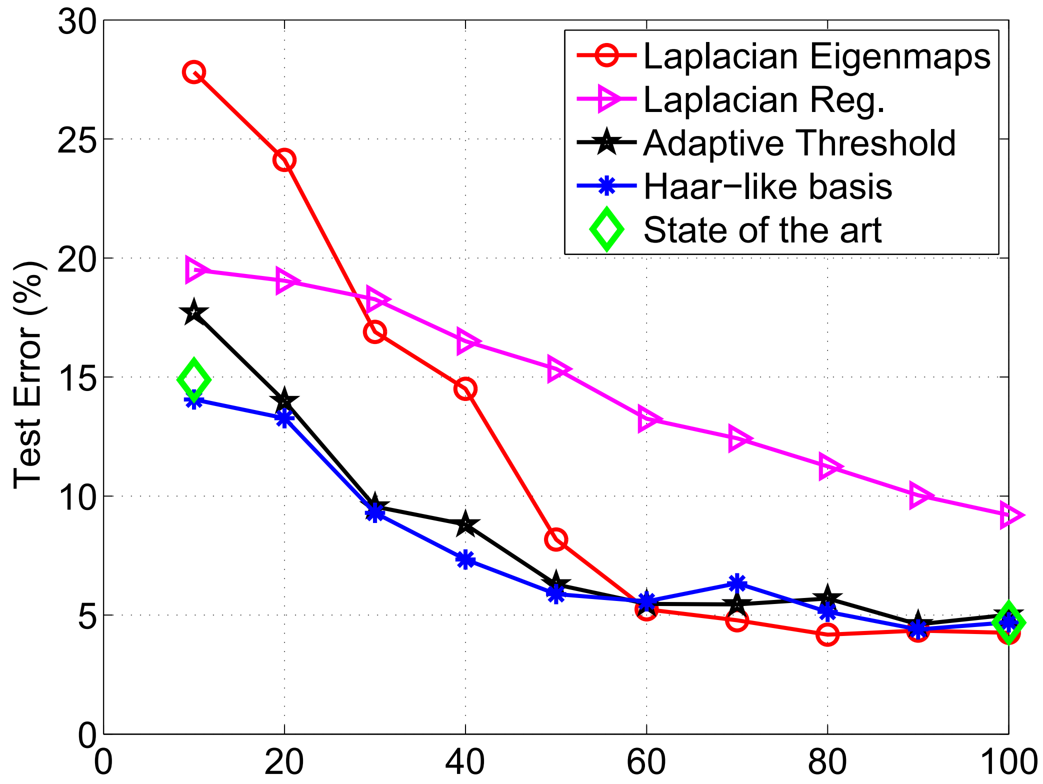
Preliminary Results



2-class USPS

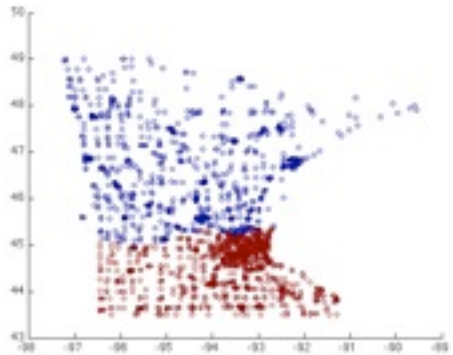
Simulation results from Gavish et al, ICML 2010

Preliminary Results

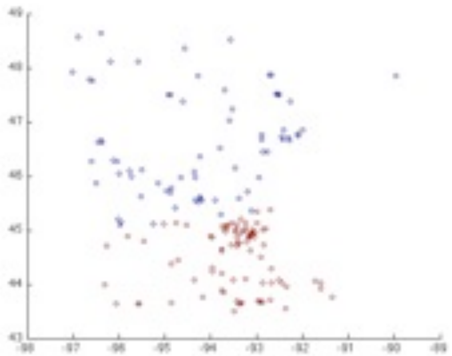


2-class USPS

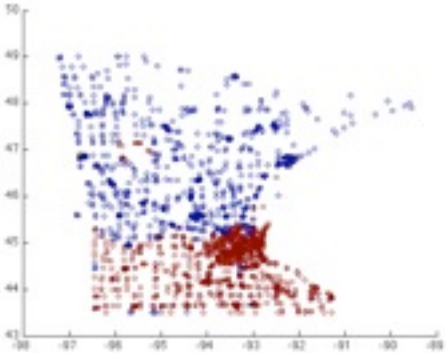
Simulation results from Gavish et al, ICML 2010



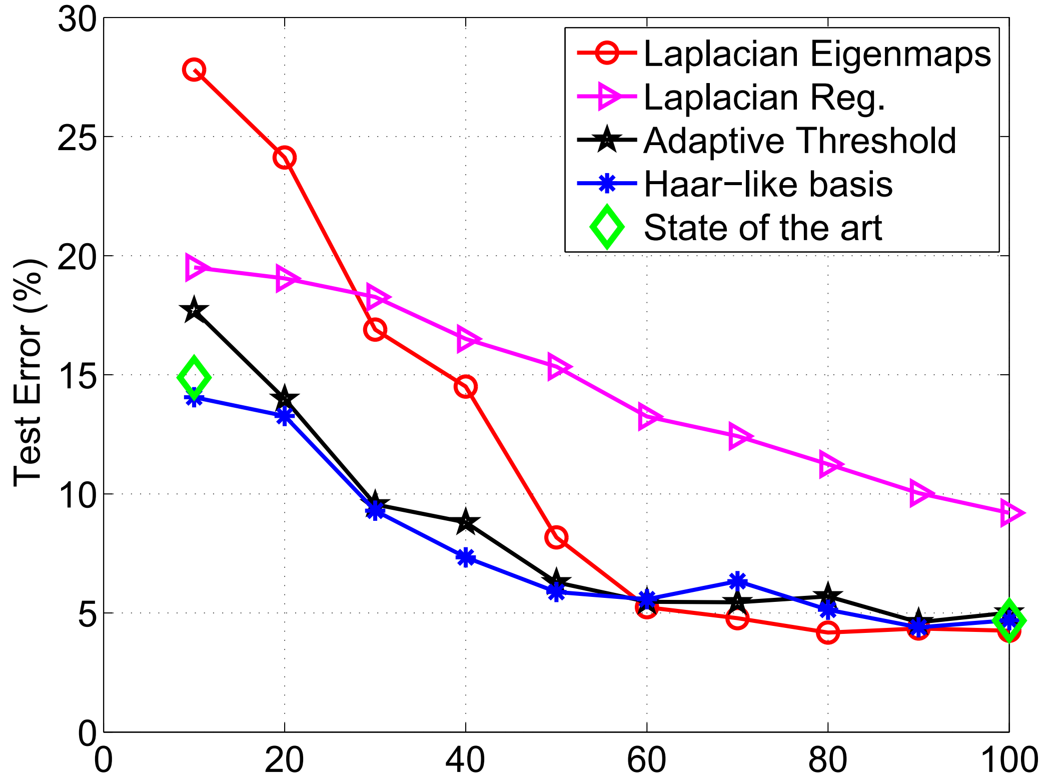
5% labeled



recovered

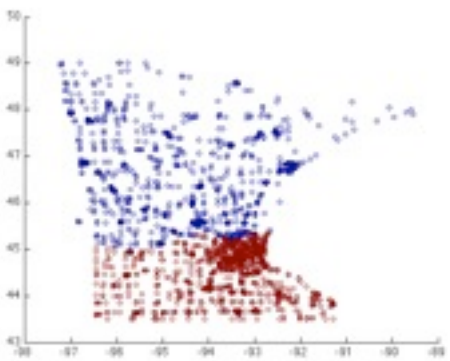


Preliminary Results

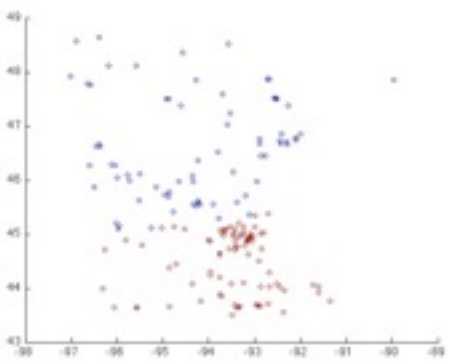


2-class USPS

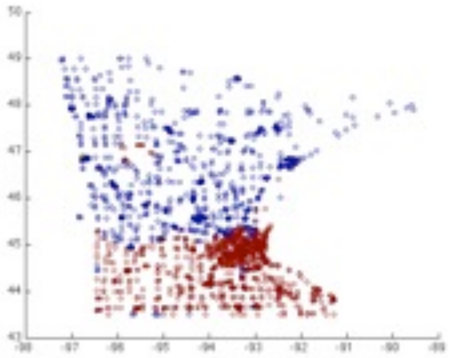
Simulation results from Gavish et al, ICML 2010



5% labeled



recovered



Is it spectacular ?

No. Comparable to state-of-art :(

Group Sparsity - take II (outlook)

Group definition too restrictive

No “spatial” (neighborhood) information

Group Sparsity - take II (outlook)

Group definition too restrictive

No “spatial” (neighborhood) information

<u>Example</u> (Composite Absolute Penalty [Mosci et al 2010, Jacob, Obozinski, Vert, 2009]):

$$\mathcal{S}(\beta) = \sum_j \gamma_j \sum_{i \in V} \sqrt{\sum_{k \sim i} \beta_{j,k}^2}$$

weights can trigger influence through scales

neighborhood of i

Group Sparsity - take II (outlook)

Group definition too restrictive

No “spatial” (neighborhood) information

Example (Composite Absolute Penalty [Mosci et al 2010, Jacob, Obozinski, Vert, 2009]):

$$\mathcal{S}(\beta) = \sum_{j} \gamma_j \sum_{i \in V} \sqrt{\sum_{k \sim i} \beta_{j,k}^2}$$

weights can trigger influence through scales

neighborhood of i

Remarks:

CAP is the composition of mixed norm and adjacency mat.

For *analysis* coefficients, at small scale $\sum_{i \in V} \sqrt{\sum_{k \sim i} \beta_{j,k}^2}$ behaves like TV

Graph wavelets

- Redundancy breaks sparsity
  - can we remove some or all of it ?
- Faster algorithms
  - traditional wavelets have fast filter banks implementation
  - whatever scale, you use the same filters
  - here: large scales -> more computations
- Goal: solve both problems at one

Kron Reduction

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

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

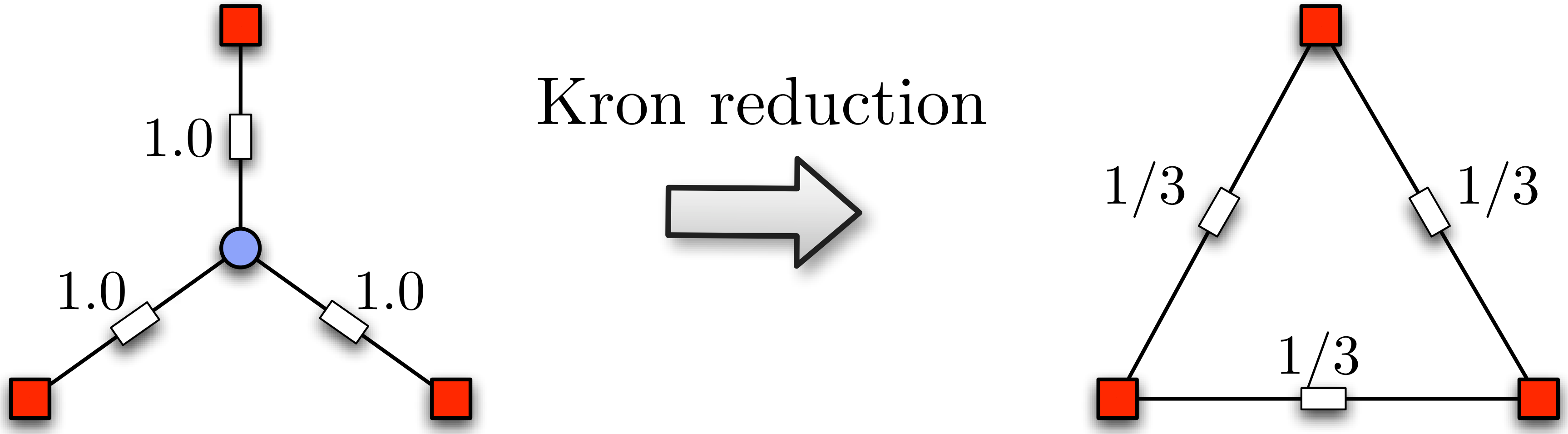
$$\mathbf{A} = \left[\begin{array}{cc} \mathbf{A}[\alpha, \alpha] & \mathbf{A}[\alpha, \alpha) \\ \mathbf{A}(\alpha, \alpha] & \mathbf{A}(\alpha, \alpha) \end{array} \right]$$

Kron Reduction

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

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

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



[Dorfler et al, 2011]

Kron Reduction

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

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

$$\mathbf{A} = \left[\begin{array}{cc} \mathbf{A}[\alpha,\alpha] & \mathbf{A}[\alpha,\alpha) \\ \mathbf{A}(\alpha,\alpha] & \mathbf{A}(\alpha,\alpha) \end{array}\right]$$

Properties: maps a weighted undirected laplacian to a weighted undirected laplacian

spectral interlacing (spectrum does not degenerate)

$$\lambda_k(\mathbf{A}) \leq \lambda_k(\mathbf{A}_r) \leq \lambda_{k+n-|\alpha|}(\mathbf{A})$$

disconnected vertices linked in reduced graph IFF there is a path that runs only through eliminated nodes

Example

Note: For a k-regular bipartite graph

$$\mathbf{L} = \begin{bmatrix} k\mathbf{I}_n & -\mathbf{A} \\ -\mathbf{A}^T & k\mathbf{I}_n \end{bmatrix}$$

Kron-reduced Laplacian: $\mathbf{L}_r = k^2\mathbf{I}_n - \mathbf{A}\mathbf{A}^T$

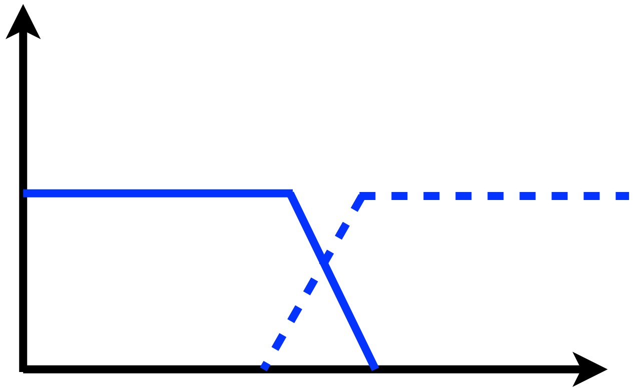
Example

Note: For a k-regular bipartite graph

$$\mathbf{L} = \begin{bmatrix} k\mathbf{I}_n & -\mathbf{A} \\ -\mathbf{A}^T & k\mathbf{I}_n \end{bmatrix}$$

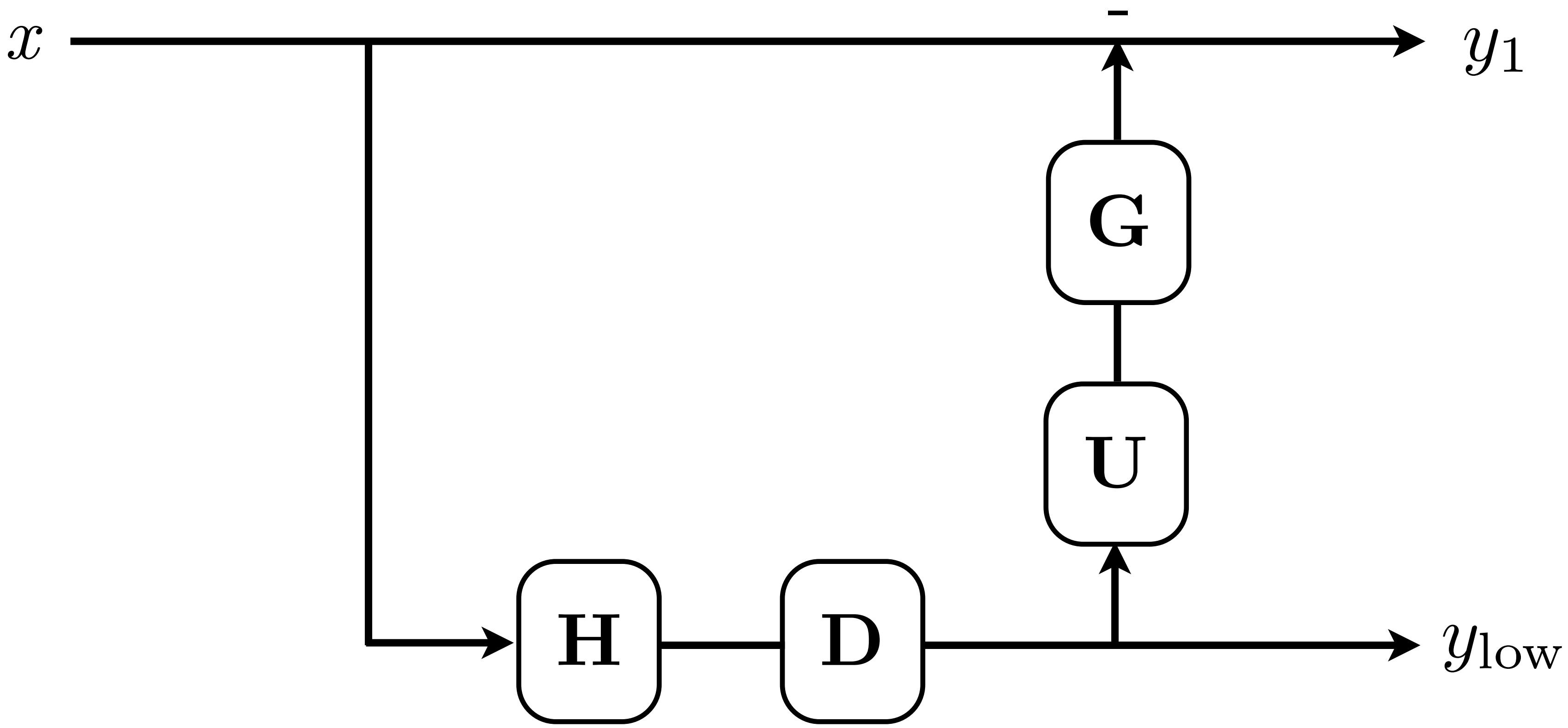
Kron-reduced Laplacian: $\mathbf{L}_r = k^2\mathbf{I}_n - \mathbf{A}\mathbf{A}^T$

$$\hat{f}_r(i) = \hat{f}(i) + \hat{f}(N-i) \quad i = 1, ..., N/2$$



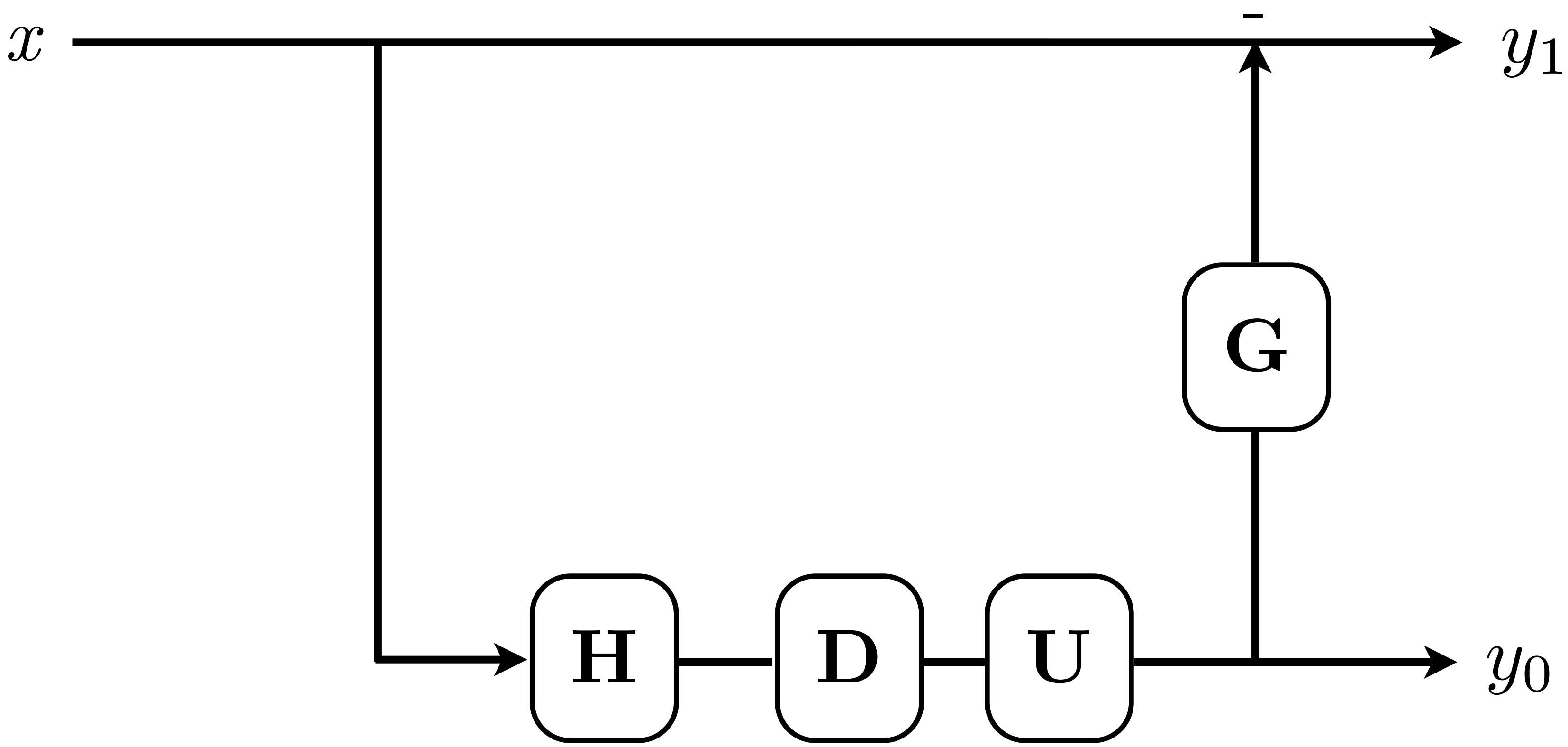
The Laplacian Pyramid

Analysis operator



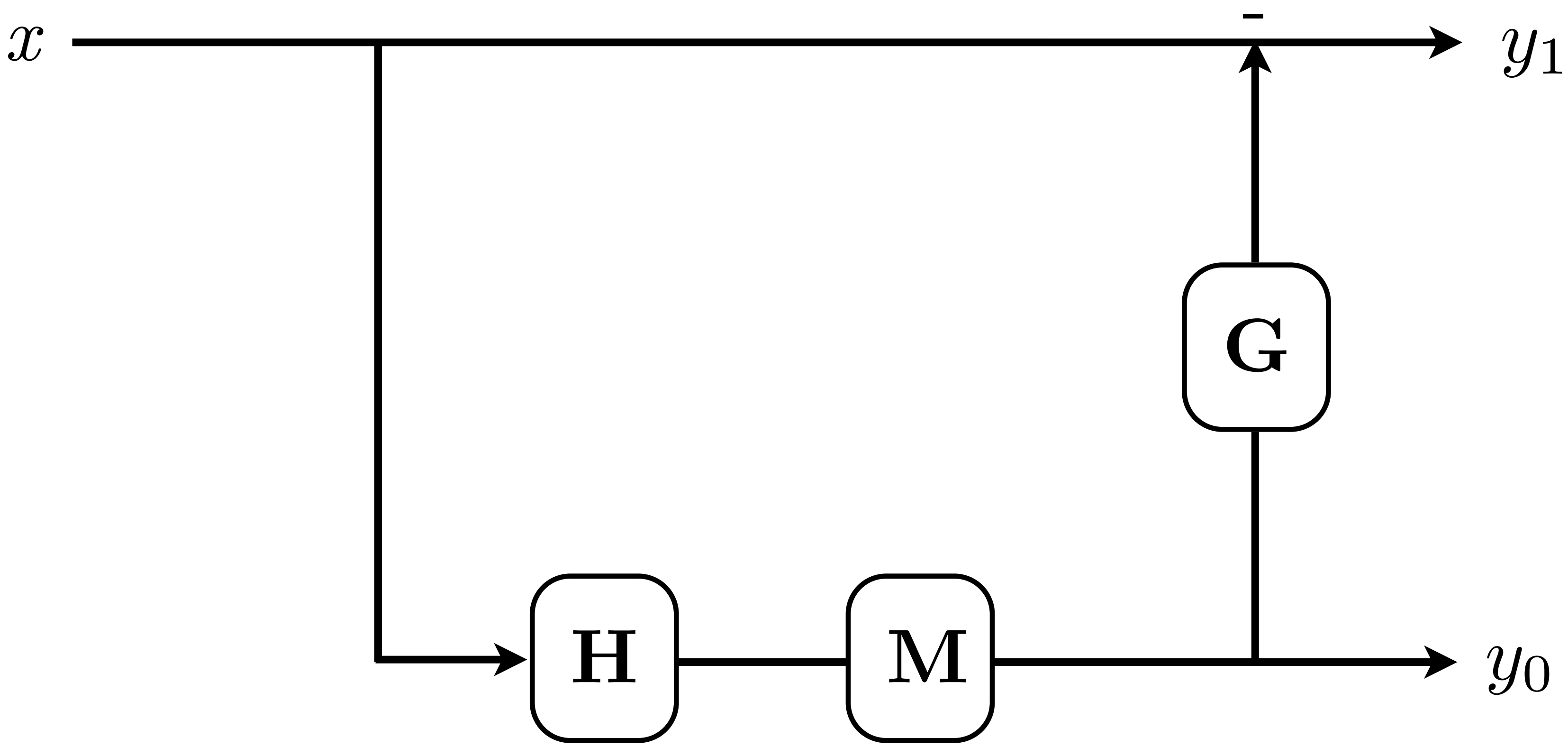
The Laplacian Pyramid

Analysis operator



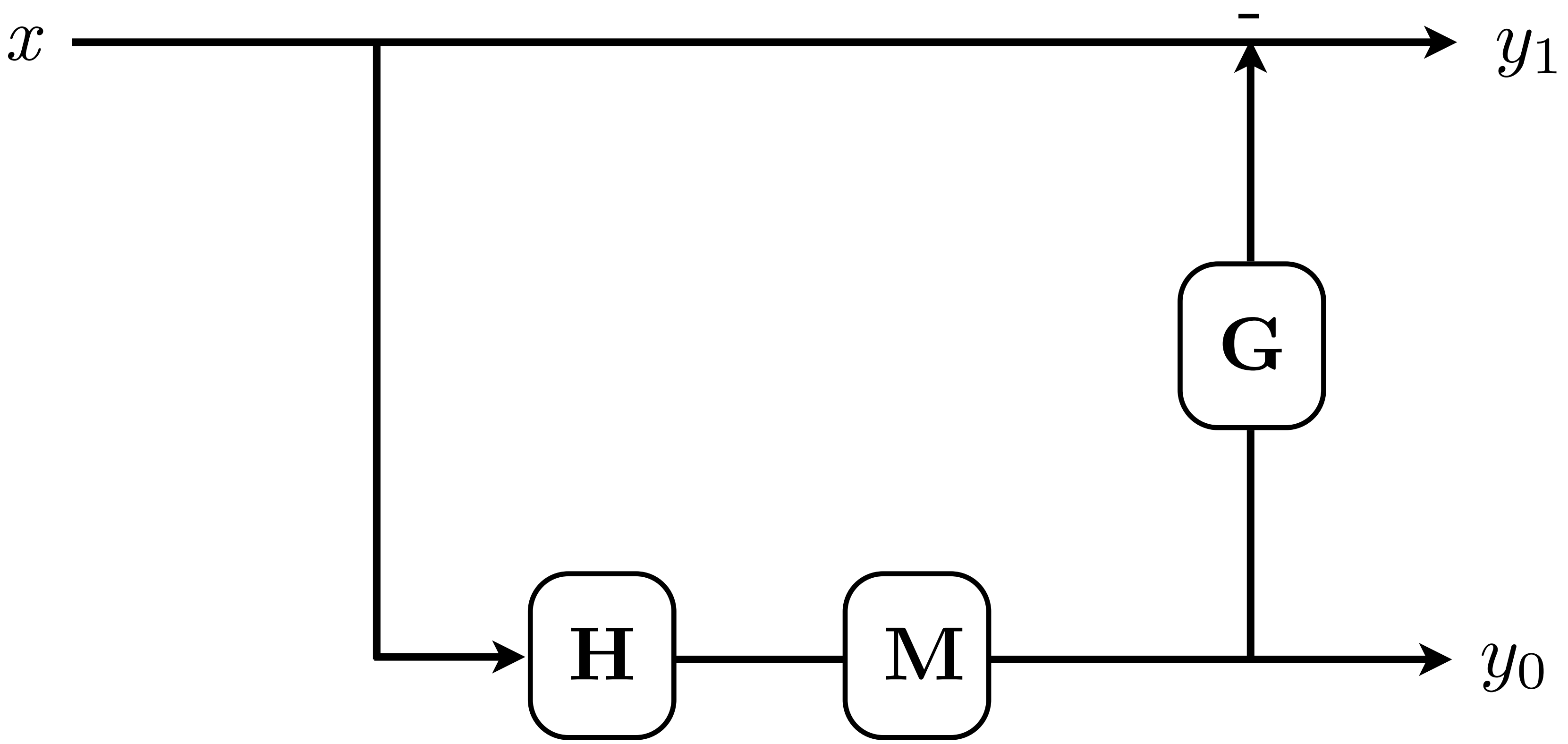
The Laplacian Pyramid

Analysis operator



The Laplacian Pyramid

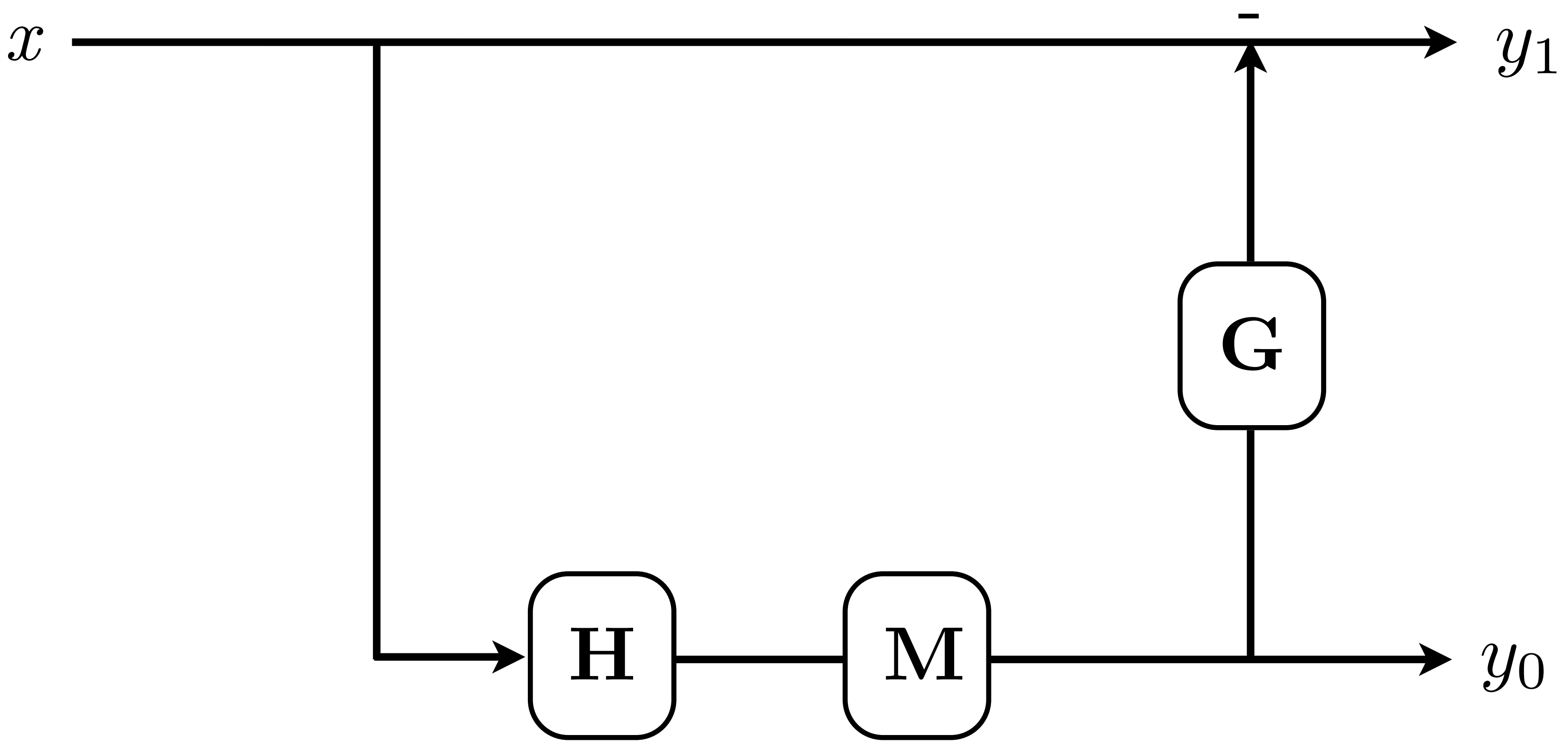
Analysis operator



$$\begin{aligned} y_0 &= \mathbf{H_m}x \\ &= \mathbf{MH}x \end{aligned} \qquad \begin{aligned} y_1 &= x - \mathbf{G}y_0 \\ &= x - \mathbf{GH_m}x \end{aligned}$$

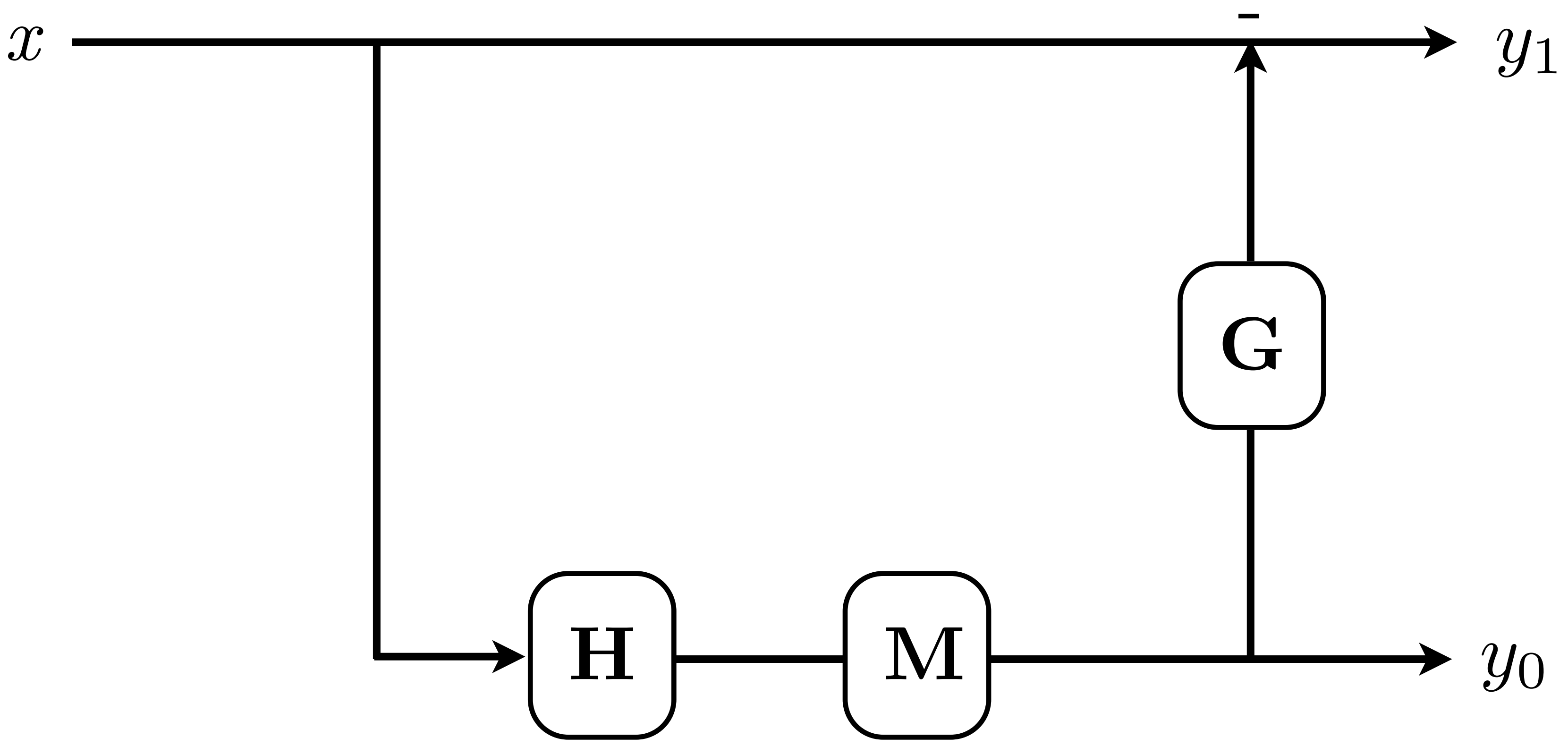
The Laplacian Pyramid

Analysis operator



The Laplacian Pyramid

Analysis operator



$$\underbrace{\begin{pmatrix} y_0 \\ y_1 \end{pmatrix}}_{y} = \underbrace{\begin{pmatrix} \mathbf{H_m} \\ \mathbf{I - GH_m} \end{pmatrix}}_{\mathbf{T_a}} x,$$

The Laplacian Pyramid

Analysis operator

$$\underbrace{\begin{pmatrix} y_0 \\ y_1 \end{pmatrix}}_{y} = \underbrace{\begin{pmatrix} \mathbf{H_m} \\ \mathbf{I} - \mathbf{G}\mathbf{H_m} \end{pmatrix}}_{\mathbf{T_a}} x,$$

The Laplacian Pyramid

Analysis operator

$$\underbrace{\begin{pmatrix} y_0 \\ y_1 \end{pmatrix}}_{y} = \underbrace{\begin{pmatrix} \mathbf{H_m} \\ \mathbf{I} - \mathbf{G}\mathbf{H_m} \end{pmatrix}}_{\mathbf{T_a}} x,$$

Simple (traditional) left inverse

$$\hat{x} = \underbrace{\begin{pmatrix} \mathbf{G} & \mathbf{I} \end{pmatrix}}_{\mathbf{T_s}} \underbrace{\begin{pmatrix} y_0 \\ y_1 \end{pmatrix}}_{y}$$

$\mathbf{T_s}\mathbf{T_a} = \mathbf{I}$ with no conditions on **H** or **G**

The Laplacian Pyramid

Pseudo Inverse ?

$${\mathbf{T_a}}^{\dagger} = \left({\mathbf{T_a}}^T \mathbf{T_a}\right)^{-1} {\mathbf{T_a}}^T$$

Let's try to use only filters

The Laplacian Pyramid

Pseudo Inverse ?

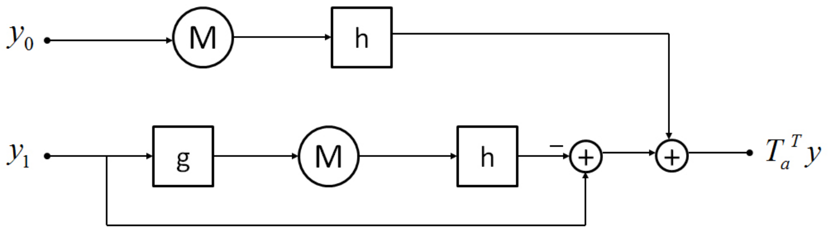
$$\mathbf{T_a}^{\dagger} = \left(\mathbf{T_a}^T\mathbf{T_a}\right)^{-1}\mathbf{T_a}^T$$

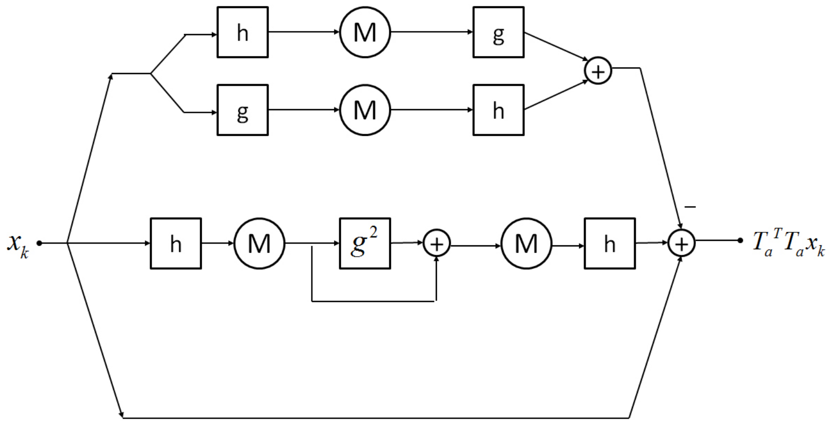
Let's try to use only filters

Define iteratively, through descent on LS:

$$\arg\min_x \|\mathbf{T_a}x - y\|_2^2 \longrightarrow \hat{x}_{k+1} = \hat{x}_k + \tau\mathbf{T_a}^T(y - \mathbf{T_a}\hat{x}_k)$$

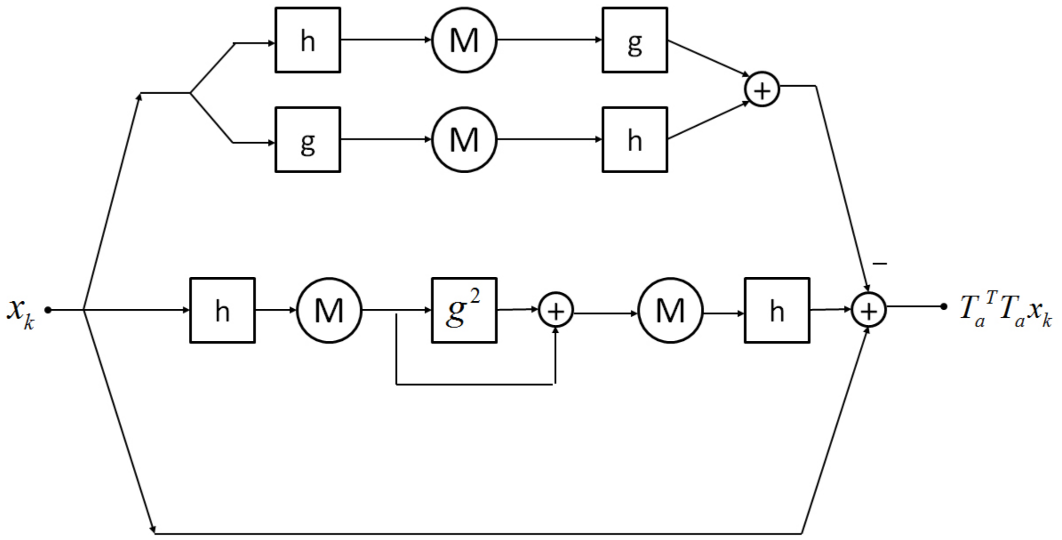
$$\mathbf{T_a}^T = (\mathbf{H_m}^T \quad \mathbf{I} - \mathbf{H_m}^T\mathbf{G}^T)$$





The Laplacian Pyramid

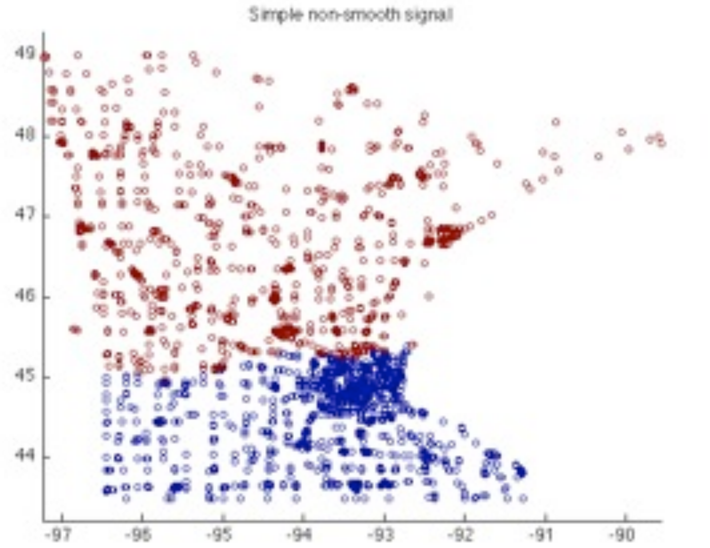
we can easily implement $\mathbf{T_a}^T\mathbf{T_a}$ with filters and masks:

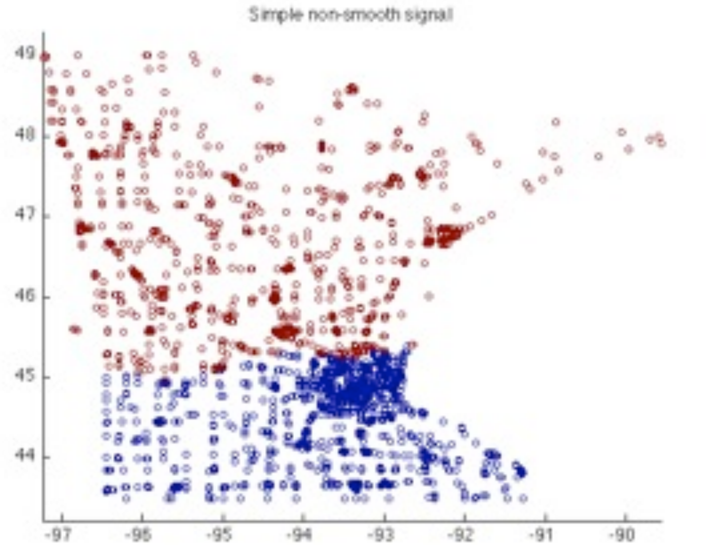


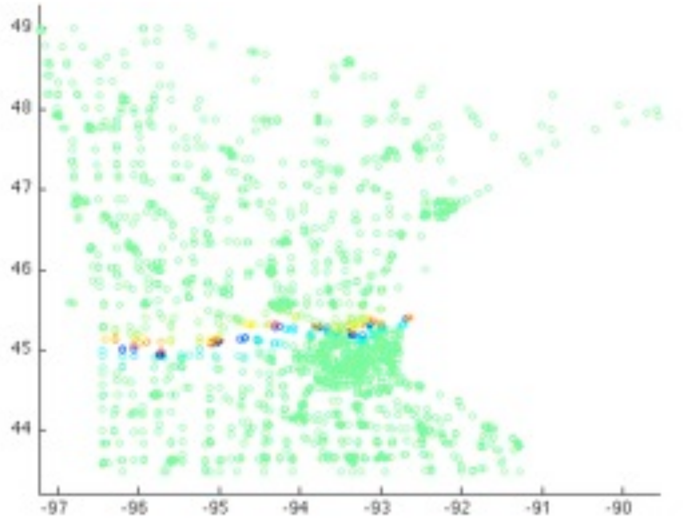
With the real symmetric matrix $\mathbf{Q} = \mathbf{T_a}^T\mathbf{T_a}$ and $b = \mathbf{T_a}^T y$

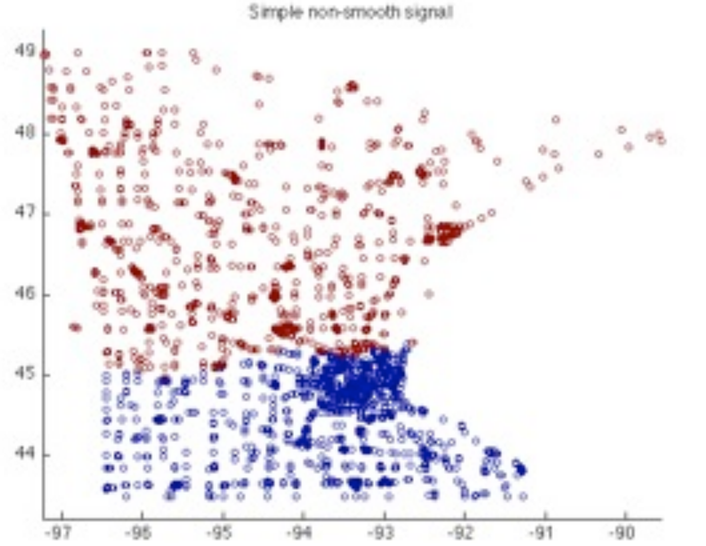
$$x_N = \tau \sum_{j=0}^{N-1} (\mathbf{I} - \tau\mathbf{Q})^j b$$

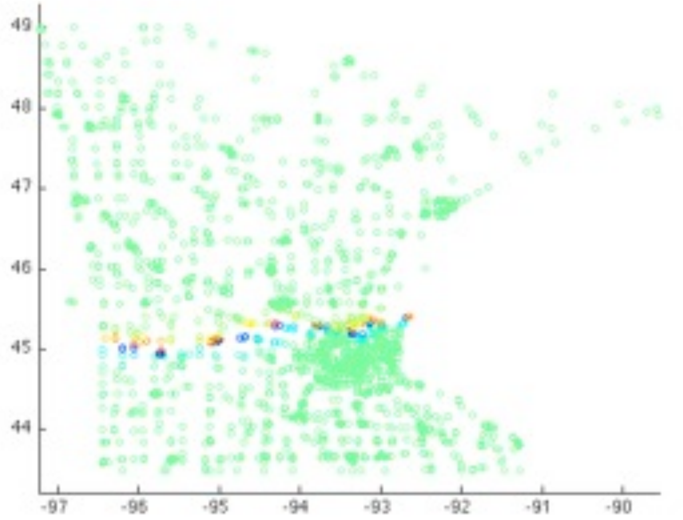
Use Chebyshev approximation of: $L(\omega) = \tau \sum_{j=0}^{N-1} (1 - \tau\omega)^j$

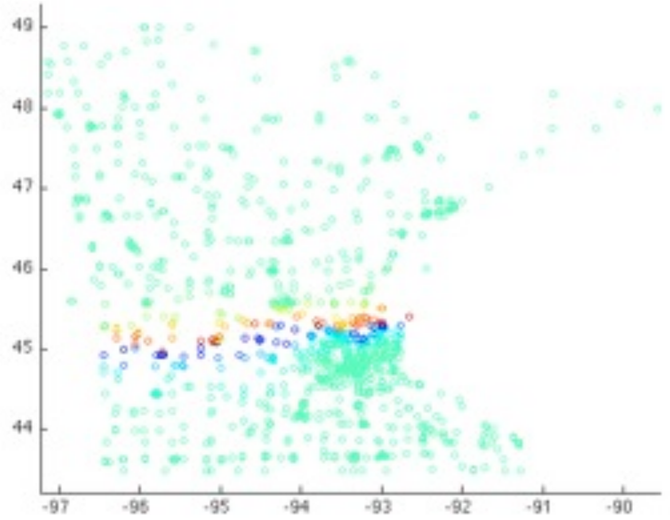

Simple non-smooth signal
49
48
47
46
45
44
-97
-96
-95
-94
-93
-92
-91
-90

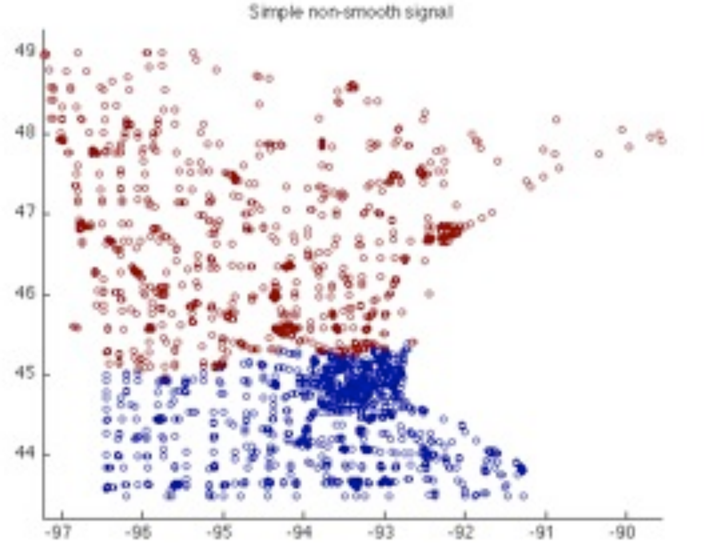

Simple non-smooth signal
49
48
47
46
45
44
-97
-96
-95
-94
-93
-92
-91
-90

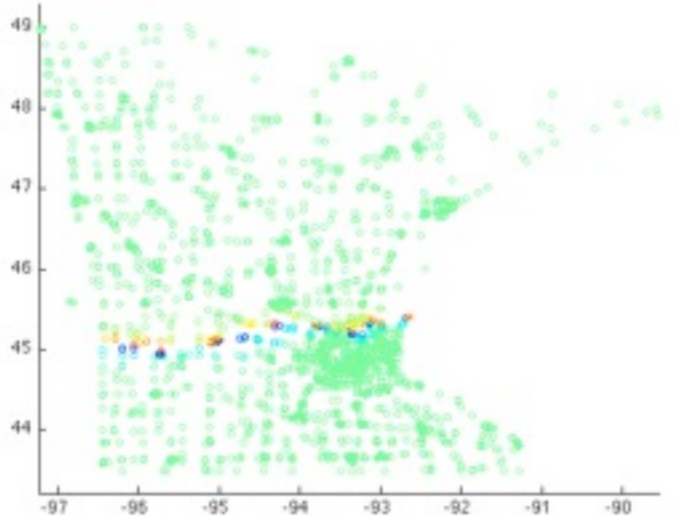

49
48
47
46
45
44
-97
-96
-95
-94
-93
-92
-91
-90

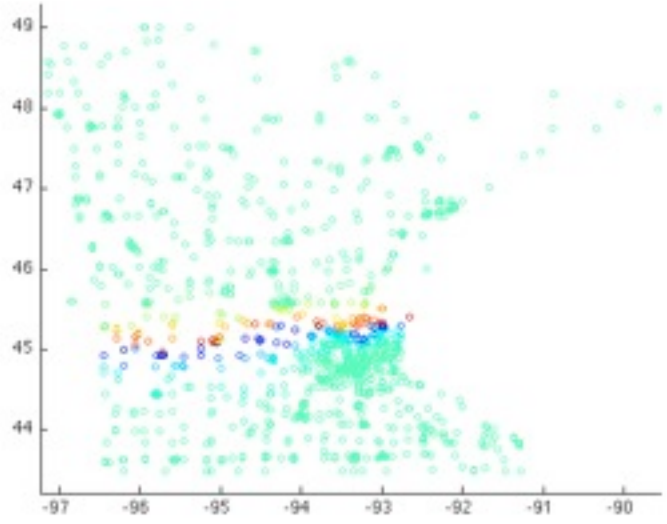

Simple non-smooth signal
49
48
47
46
45
44
-97
-96
-95
-94
-93
-92
-91
-90

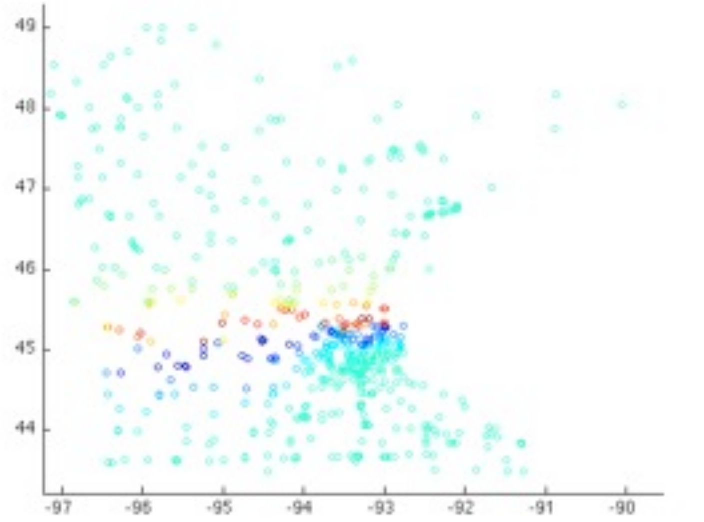

49
48
47
46
45
44
-97
-96
-95
-94
-93
-92
-91
-90

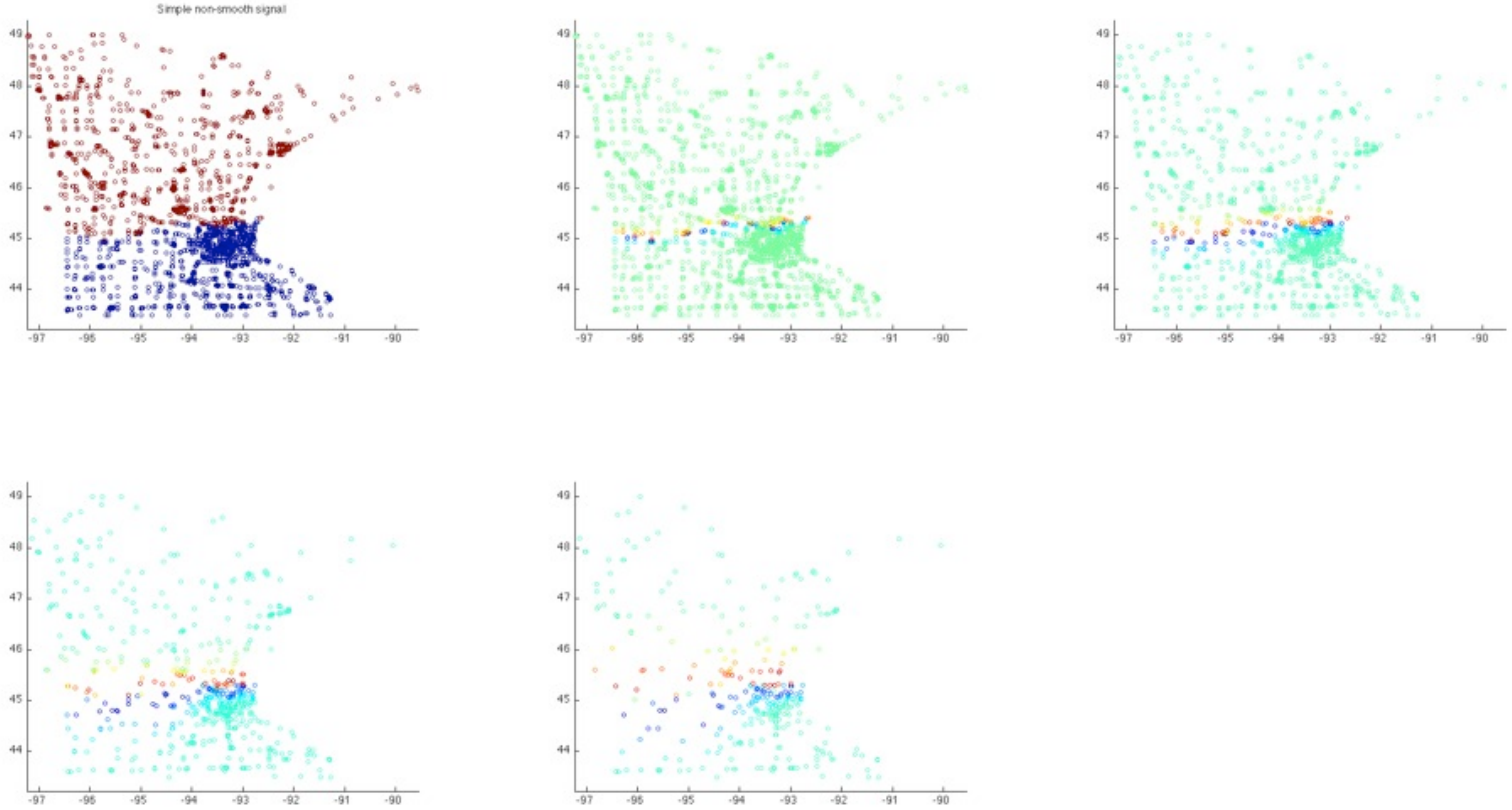

49
48
47
46
45
44
-97
-96
-95
-94
-93
-92
-91
-90

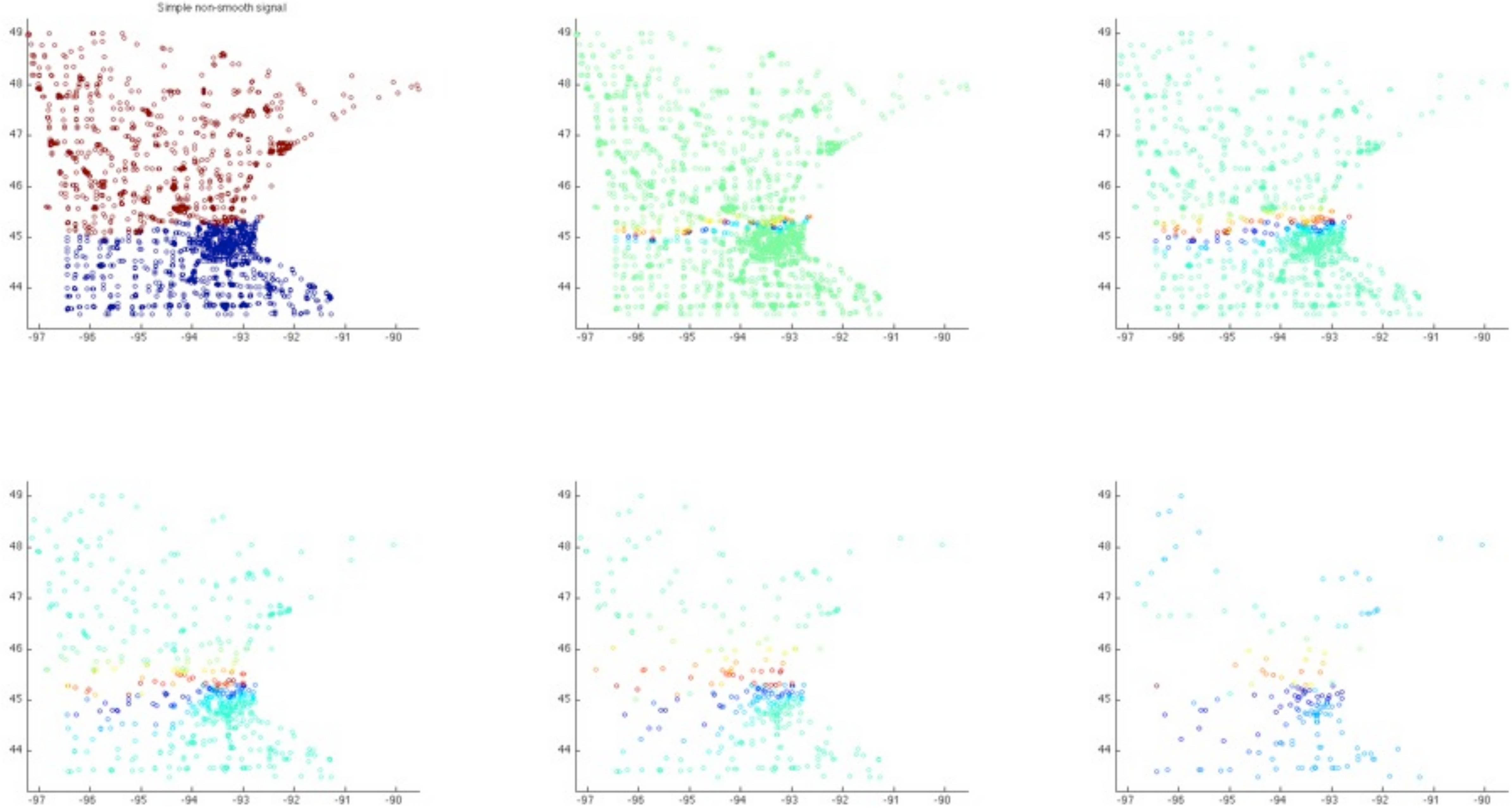

Simple non-smooth signal
49
48
47
46
45
44
-97
-96
-95
-94
-93
-92
-91
-90


49
48
47
46
45
44
-97
-96
-95
-94
-93
-92
-91
-90


49
48
47
46
45
44
-97
-96
-95
-94
-93
-92
-91
-90

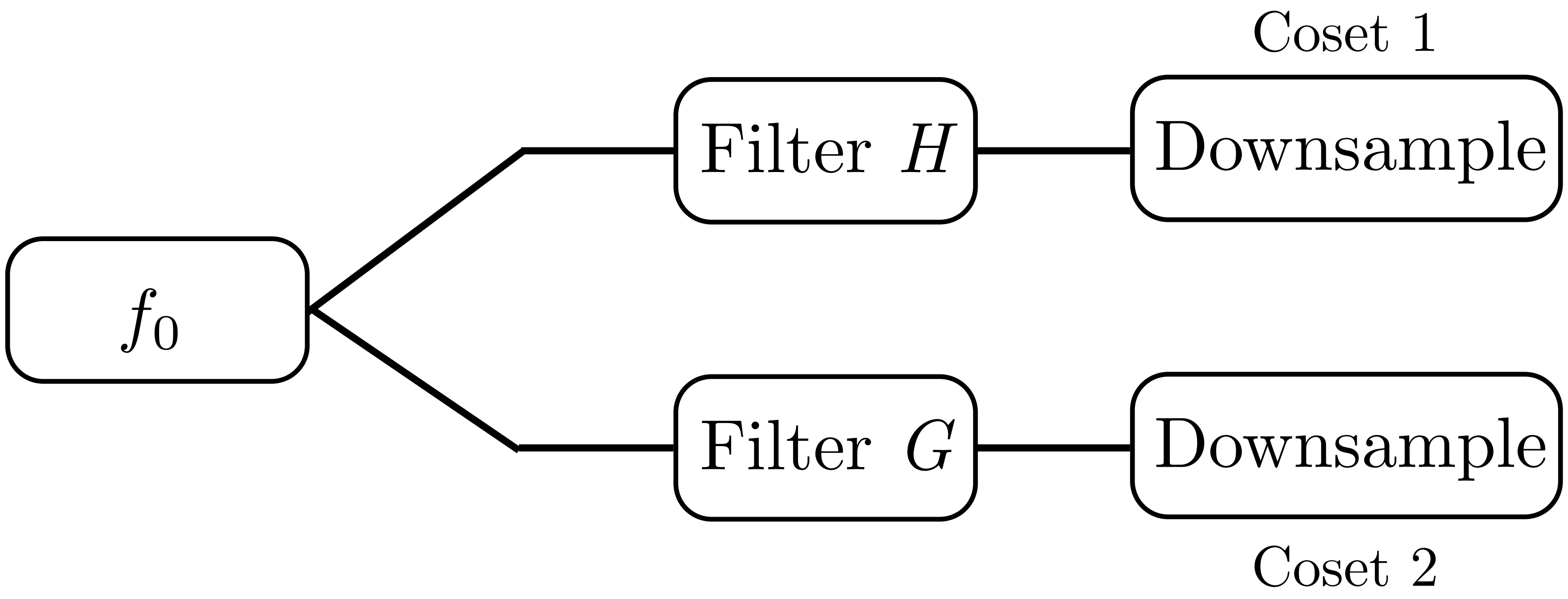

49
48
47
46
45
44
-97
-96
-95
-94
-93
-92
-91
-90


Simple non-smooth signal
49
48
47
46
45
44
-97
-96
-95
-94
-93
-92
-91
-90


Simple non-smooth signal
49
48
47
46
45
44
-97
-96
-95
-94
-93
-92
-91
-90

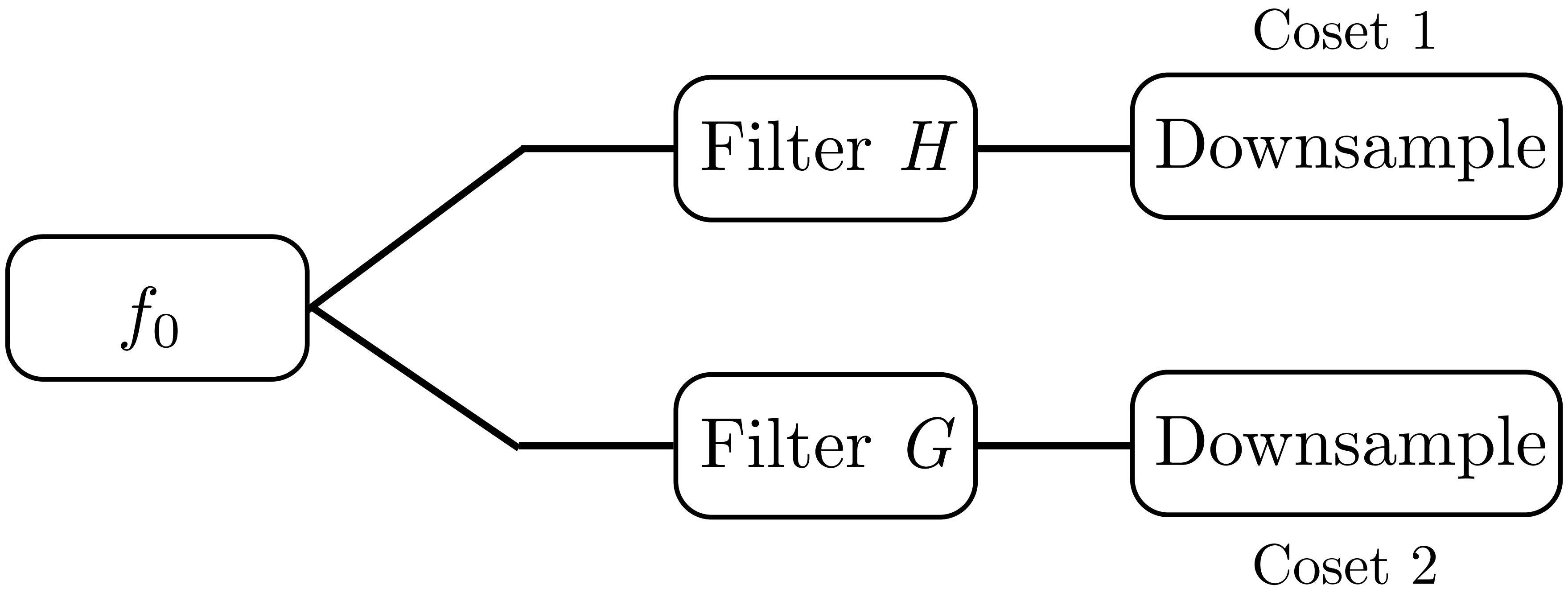
Filter Banks

2 critically sampled channels



Filter Banks

2 critically sampled channels



Theorem: For a k-RBG, the filter bank is perfect-reconstruction IFF

$$|H(i)|^2 + |G(i)|^2 = 2$$

$$H(i)G(N-i) + H(N-i)G(i) = 0$$

Outline

1 Introduction

2 Spectral Graph Theory Background

3 Wavelet Constructions on Graphs

4 Approximate Graph Multiplier Operators

5 Distributed Signal Processing via the Chebyshev Approximation

6 Open Issues and Challenges

Chebyshev Polynomials

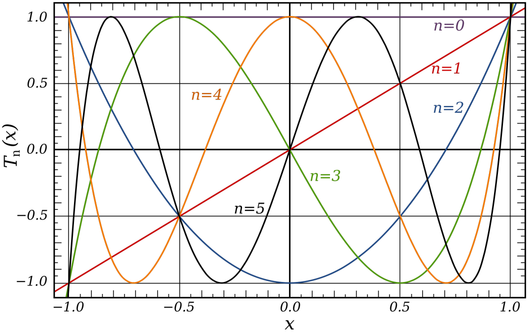
- $T_n(x) := \cos\big(n \arccos(x)\big),$
  $x \in [-1, 1],$
  $n = 0, 1, 2, \ldots$

- $T_0(x) = 1$

  $T_1(x) = x$

  $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$

  for $k \geq 2$



Source: Wikipedia.

Chebyshev Polynomial Expansion and Approximation

- Chebyshev polynomials form an orthogonal basis for $L^2\left([-1,1], \frac{dx}{\sqrt{1-x^2}}\right)$
  - Every $h \in L^2\left([-1,1], \frac{dx}{\sqrt{1-x^2}}\right)$ can be represented as

$$h(x) = \frac{1}{2}c_0 + \sum_{k=1}^{\infty} c_k T_k(x), \text{ where } c_k = \frac{2}{\pi}\int_0^{\pi} \cos(k\theta)h(\cos(\theta))d\theta$$

Chebyshev Polynomial Expansion and Approximation

- Chebyshev polynomials form an orthogonal basis for $L^2\left([-1,1], \frac{dx}{\sqrt{1-x^2}}\right)$
  - Every $h \in L^2\left([-1,1], \frac{dx}{\sqrt{1-x^2}}\right)$ can be represented as

$$h(x) = \frac{1}{2}c_0 + \sum_{k=1}^{\infty} c_k T_k(x), \text{ where } c_k = \frac{2}{\pi}\int_0^{\pi} \cos(k\theta) h(\cos(\theta)) d\theta$$

- K^{th} order Chebyshev approximation to a continuous function on an interval provides a near-optimal approximation (in the sup norm) amongst all polynomials of degree K

Chebyshev Polynomial Expansion and Approximation

- Chebyshev polynomials form an orthogonal basis for $L^2\left([-1,1], \frac{dx}{\sqrt{1-x^2}}\right)$
  - Every $h \in L^2\left([-1,1], \frac{dx}{\sqrt{1-x^2}}\right)$ can be represented as

$$h(x) = \frac{1}{2}c_0 + \sum_{k=1}^{\infty} c_k T_k(x), \text{ where } c_k = \frac{2}{\pi}\int_0^{\pi} \cos(k\theta)h(\cos(\theta))d\theta$$

- K^{th} order Chebyshev approximation to a continuous function on an interval provides a near-optimal approximation (in the sup norm) amongst all polynomials of degree K

Shifted Chebyshev Polynomials

- To shift the domain from [-1,1] to [0,A], define

$$\overline{T}_k(x) := T_k\left(\frac{x}{\alpha} - 1\right), \text{ where } \alpha := \frac{A}{2}$$

- $\overline{T}_k(x) = \frac{2}{\alpha}(x-\alpha)\overline{T}_{k-1}(x) - \overline{T}_{k-2}(x)$ for $k \geq 2$

Fast Chebyshev Approx. of a Graph Multiplier Operator

Let $\Phi \in \mathbb{R}^{N \times N}$ be a graph Fourier multiplier with $\Phi f = \begin{bmatrix} (\Phi f)_1 \\ \vdots \\ (\Phi f)_N \end{bmatrix}$

Approximate Graph Fourier Multiplier Operator

$$\begin{aligned}
\left(\Phi f\right)_n = \sum_{\ell=0}^{N-1} g(\lambda_\ell)\hat{f}(\ell)\chi_\ell(n) &= \sum_{\ell=0}^{N-1} \left[\frac{1}{2}c_0 + \sum_{k=1}^{\infty} c_k \overline{T}_k(\lambda_\ell)\right] \hat{f}(\ell)\chi_\ell(n) \\
&\approx \sum_{\ell=0}^{N-1} \left[\frac{1}{2}c_0 + \sum_{k=1}^{K} c_k \overline{T}_k(\lambda_\ell)\right] \hat{f}(\ell)\chi_\ell(n) \\
&= \left(\frac{1}{2}c_0 f + \sum_{k=1}^{K} c_k \overline{T}_k(\mathcal{L}) f\right)_n := \left(\tilde{\Phi} f\right)_n
\end{aligned}$$

Here, $\overline{T}_k(\mathcal{L}) \in \mathbb{R}^{N \times N}$ and $\left(\overline{T}_k(\mathcal{L})f\right)_n := \sum_{\ell=0}^{N-1} \overline{T}_k(\lambda_\ell)\hat{f}(\ell)\chi_\ell(n)$

Fast Chebyshev Approx. of a Graph Fourier Multiplier

$$\tilde{\Phi} f = \frac{1}{2} c_0 f + \sum_{k=1}^{K} c_k \overline{T}_k(\mathcal{L}) f \approx \Phi f$$

Question: Why do we call this a fast approximation?

Fast Chebyshev Approx. of a Graph Fourier Multiplier

$$\tilde{\Phi} f = \tfrac{1}{2} c_0 f + \sum_{k=1}^{K} c_k \overline{T}_k(\mathcal{L}) f \approx \Phi f$$

Question: Why do we call this a fast approximation?

Answer: From the Chebyshev polynomial recursion property, we have:

$$\begin{aligned}
\overline{T}_0(\mathcal{L}) f &= f \\
\overline{T}_1(\mathcal{L}) f &= \frac{1}{\alpha} \mathcal{L} f - f, \quad \text{where } \alpha := \frac{\lambda_{\max}}{2} \\
\overline{T}_k(\mathcal{L}) f &= \frac{2}{\alpha} (\mathcal{L} - \alpha I) \left(\overline{T}_{k-1}(\mathcal{L}) f \right) - \overline{T}_{k-2}(\mathcal{L}) f \\
&= \frac{2}{\alpha} \mathcal{L} \overline{T}_{k-1}(\mathcal{L}) f - 2 \overline{T}_{k-1}(\mathcal{L}) f - \overline{T}_{k-2}(\mathcal{L}) f
\end{aligned}$$

Fast Chebyshev Approx. of a Graph Fourier Multiplier

$$\tilde{\Phi}f = \tfrac{1}{2}c_0 f + \sum_{k=1}^{K} c_k \overline{T}_k(\mathcal{L})f \approx \Phi f$$

Question: Why do we call this a fast approximation?

Answer: From the Chebyshev polynomial recursion property, we have:

$$\begin{aligned}
\overline{T}_0(\mathcal{L})f &= f \\
\overline{T}_1(\mathcal{L})f &= \frac{1}{\alpha}\mathcal{L}f - f, \quad \text{where } \alpha := \frac{\lambda_{\max}}{2} \\
\overline{T}_k(\mathcal{L})f &= \frac{2}{\alpha}(\mathcal{L} - \alpha I)\left(\overline{T}_{k-1}(\mathcal{L})f\right) - \overline{T}_{k-2}(\mathcal{L})f \\
&= \frac{2}{\alpha}\mathcal{L}\overline{T}_{k-1}(\mathcal{L})f - 2\overline{T}_{k-1}(\mathcal{L})f - \overline{T}_{k-2}(\mathcal{L})f
\end{aligned}$$

- Does not require explicit computation of the eigenvectors of the Laplacian
- Computational cost proportional to # nonzero entries in the Laplacian
- This corresponds to the number of edges in the communication graph
- Large, sparse graph $\Rightarrow$ $\tilde{\Phi}f$ far more efficient than Φf

Approximation Error

- Let $\mathbf{\Phi}$ be a union of η generalized graph multiplier operators:

$$\mathbf{\Phi} = [\mathbf{\Psi}_1; \mathbf{\Psi}_2; \ldots; \mathbf{\Psi}_\eta], \text{ where } \mathbf{\Psi}_j = \sum_{\ell=0}^{N-1} g_j(\lambda_\ell) \boldsymbol{\chi}_\ell \boldsymbol{\chi}_\ell^*$$

Approximation Error

- Let $\mathbf{\Phi}$ be a union of η generalized graph multiplier operators:

$$\mathbf{\Phi} = [\mathbf{\Psi}_1; \mathbf{\Psi}_2; \ldots; \mathbf{\Psi}_\eta], \text{ where } \mathbf{\Psi}_j = \sum_{\ell=0}^{N-1} g_j(\lambda_\ell)\boldsymbol{\chi}_\ell\boldsymbol{\chi}_\ell^*$$

- Define $B(K) := \max_{j=1,2,\ldots,\eta} \left\{ \sup_{\lambda\in[0,\lambda_{\max}]} \left\{ \left| g_j(\lambda) - p_j^K(\lambda) \right| \right\} \right\}$

Approximation Error

- Let $\mathbf{\Phi}$ be a union of η generalized graph multiplier operators:

$$\mathbf{\Phi} = [\mathbf{\Psi}_1; \mathbf{\Psi}_2; \ldots; \mathbf{\Psi}_\eta], \text{ where } \mathbf{\Psi}_j = \sum_{\ell=0}^{N-1} g_j(\lambda_\ell) \boldsymbol{\chi}_\ell \boldsymbol{\chi}_\ell^*$$

- Define $B(K) := \max_{j=1,2,\ldots,\eta} \left\{ \sup_{\lambda \in [0,\lambda_{\max}]} \left\{ \left| g_j(\lambda) - p_j^K(\lambda) \right| \right\} \right\}$

Proposition

$$\|\mathbf{\Phi} - \tilde{\mathbf{\Phi}}\|_2 := \max_{\mathbf{f} \neq \mathbf{0}} \frac{\|(\mathbf{\Phi} - \tilde{\mathbf{\Phi}})\mathbf{f}\|_2}{\|\mathbf{f}\|_2} \leq B(K)\sqrt{\eta N}.$$

Approximation Error

- Let $\mathbf{\Phi}$ be a union of η generalized graph multiplier operators:

$$\mathbf{\Phi} = [\mathbf{\Psi}_1; \mathbf{\Psi}_2; \ldots; \mathbf{\Psi}_\eta], \text{ where } \mathbf{\Psi}_j = \sum_{\ell=0}^{N-1} g_j(\lambda_\ell)\boldsymbol{\chi}_\ell \boldsymbol{\chi}_\ell^*$$

- Define $B(K) := \max_{j=1,2,\ldots,\eta} \left\{ \sup_{\lambda \in [0,\lambda_{\max}]} \left\{ \left| g_j(\lambda) - p_j^K(\lambda) \right| \right\} \right\}$

Proposition

$$\|\mathbf{\Phi} - \tilde{\mathbf{\Phi}}\|_2 := \max_{\mathbf{f} \neq \mathbf{0}} \frac{\|(\mathbf{\Phi} - \tilde{\mathbf{\Phi}})\mathbf{f}\|_2}{\|\mathbf{f}\|_2} \leq B(K)\sqrt{\eta N}.$$

Proposition (see, e.g., Mason and Handscomb, 2003)

If $g_j(\cdot)$ has $M+1$ continuous derivatives for all j, then $B(K) = \mathcal{O}\left(K^{-M}\right)$.

Outline

1 Introduction

2 Spectral Graph Theory Background

3 Wavelet Constructions on Graphs

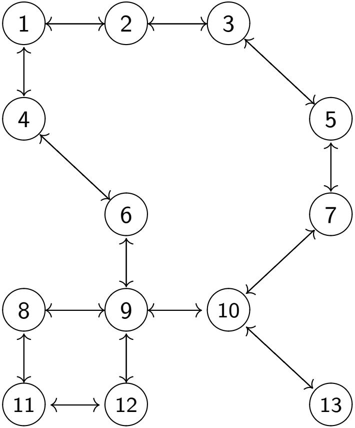
4 Approximate Graph Multiplier Operators

5 Distributed Signal Processing via the Chebyshev Approximation

6 Open Issues and Challenges

Motivating Application: Distributed Denoising

- Sensor network with N sensors
- Noisy signal in $\mathbb{R}^N$: $y = x +$ noise
- Node n only observes y_n and wants to estimate x_n
- No central entity - nodes can only send messages to their neighbors in the communication graph
- However, communication is costly
- Prior info, e.g., signal is smooth or piecewise smooth w.r.t. graph structure
    - If two sensors are close enough to communicate, their observations are more likely to be correlated



Distributed Computation

$$\left(\tilde{\Phi} f\right)_n = \left(\frac{1}{2}c_0 f + \sum_{k=1}^{K} c_k \overline{T}_k(\mathcal{L}) f\right)_n$$

NODE n'S KNOWLEDGE:

1. $(f)_n$
2. Neighbors and weights of edges to its neighbors
3. Graph Fourier multiplier $g(\cdot)$, which is used to compute $c_o, c_1, \ldots, c_K$
4. Loose upper bound on $\lambda_{\max}$

Distributed Computation

$$\left(\tilde{\Phi} f\right)_n = \left(\frac{1}{2}c_0 f + \sum_{k=1}^{K} c_k \overline{T}_k(\mathcal{L}) f\right)_n$$

NODE n'S KNOWLEDGE:

1. $(f)_n$
2. Neighbors and weights of edges to its neighbors
3. Graph Fourier multiplier $g(\cdot)$, which is used to compute $c_o, c_1, \ldots, c_K$
4. Loose upper bound on $\lambda_{\max}$

Task: Compute $(\overline{T}_k(\mathcal{L})f)_n,\ k \in \{1, 2, \ldots, K\}$ in a distributed manner

Distributed Computation

$$\left(\tilde{\Phi} f\right)_n = \left(\frac{1}{2} c_0 f + \sum_{k=1}^{K} c_k \overline{T}_k(\mathcal{L}) f\right)_n$$

Node n's knowledge:

1. $(f)_n$
2. Neighbors and weights of edges to its neighbors
3. Graph Fourier multiplier $g(\cdot)$, which is used to compute $c_o, c_1, \ldots, c_K$
4. Loose upper bound on $\lambda_{\max}$

Task: Compute $(\overline{T}_k(\mathcal{L})f)_n, \; k \in \{1, 2, \ldots, K\}$ in a distributed manner

- $(\overline{T}_1(\mathcal{L})f)_n = \frac{1}{\alpha}(\mathcal{L}f)_n - (f)_n = \frac{1}{\alpha} \begin{bmatrix} \\ \boxed{0\ \mathcal{L}_{n,2}\ 0\ 0\ 0\ \mathcal{L}_{n,6}\ 0\ 0\ 0} \\ \\ \end{bmatrix} \begin{bmatrix} \boxed{\mathbf{f}} \end{bmatrix} - (f)_n$

Distributed Computation

$$\left(\tilde{\Phi} f\right)_n = \left(\frac{1}{2} c_0 f + \sum_{k=1}^{K} c_k \overline{T}_k(\mathcal{L}) f\right)_n$$

NODE n'S KNOWLEDGE:

1. $(f)_n$
2. Neighbors and weights of edges to its neighbors
3. Graph Fourier multiplier $g(\cdot)$, which is used to compute $c_o, c_1, \ldots, c_K$
4. Loose upper bound on $\lambda_{\max}$

Task: Compute $(\overline{T}_k(\mathcal{L}) f)_n, \; k \in \{1, 2, \ldots, K\}$ in a distributed manner

- $(\overline{T}_1(\mathcal{L}) f)_n = \frac{1}{\alpha}(\mathcal{L} f)_n - (f)_n = \frac{1}{\alpha} \begin{pmatrix} \\ 0\ \mathcal{L}_{n,2}\ 0\ 0\ 0\ \mathcal{L}_{n,6}\ 0\ 0\ 0 \\ \\ \end{pmatrix} \begin{pmatrix} \\ \mathbf{f} \\ \\ \end{pmatrix} - (f)_n$
- $\left(\overline{T}_k(\mathcal{L}) f\right)_n = \left(\frac{2}{\alpha} \mathcal{L} \overline{T}_{k-1}(\mathcal{L}) f\right)_n - \left(2 \overline{T}_{k-1}(\mathcal{L}) f\right)_n - \left(\overline{T}_{k-2}(\mathcal{L}) f\right)_n$
- To get $(\overline{T}_2(\mathcal{L}) f)_n$, suffices to compute $(\mathcal{L} \overline{T}_1(\mathcal{L}) f)_n = \begin{pmatrix} \\ 0\ \mathcal{L}_{n,2}\ 0\ 0\ 0\ \mathcal{L}_{n,6}\ 0\ 0\ 0 \\ \\ \end{pmatrix} \begin{pmatrix} \\ \overline{T}_1(\mathcal{L})f \\ \\ \end{pmatrix}$

Distributed Computation

$$\left(\tilde{\Phi} f\right)_n = \left(\frac{1}{2}c_0 f + \sum_{k=1}^{K} c_k \overline{T}_k(\mathcal{L})f\right)_n$$

NODE n'S KNOWLEDGE:

1. $(f)_n$
2. Neighbors and weights of edges to its neighbors
3. Graph Fourier multiplier $g(\cdot)$, which is used to compute $c_o, c_1, \ldots, c_K$
4. Loose upper bound on $\lambda_{\max}$

Task: Compute $(\overline{T}_k(\mathcal{L})f)_n,\ k \in \{1, 2, \ldots, K\}$ in a distributed manner

- $(\overline{T}_1(\mathcal{L})f)_n = \frac{1}{\alpha}(\mathcal{L}f)_n - (f)_n = \frac{1}{\alpha}\begin{bmatrix} \\ 0\ \mathcal{L}_{n,2}\ 0\ 0\ 0\ \mathcal{L}_{n,6}\ 0\ 0\ 0 \\ \\ \end{bmatrix}\begin{bmatrix} \\ \mathbf{f} \\ \\ \end{bmatrix} - (f)_n$
- $\left(\overline{T}_k(\mathcal{L})f\right)_n = \left(\frac{2}{\alpha}\mathcal{L}\overline{T}_{k-1}(\mathcal{L})f\right)_n - \left(2\overline{T}_{k-1}(\mathcal{L})f\right)_n - \left(\overline{T}_{k-2}(\mathcal{L})f\right)_n$
- To get $(\overline{T}_2(\mathcal{L})f)_n$, suffices to compute $(\mathcal{L}\overline{T}_1(\mathcal{L})f)_n = \begin{bmatrix} \\ 0\ \mathcal{L}_{n,2}\ 0\ 0\ 0\ \mathcal{L}_{n,6}\ 0\ 0\ 0 \\ \\ \end{bmatrix}\begin{bmatrix} \\ \overline{\mathbf{T}}_1(\mathcal{L})\mathbf{f} \\ \\ \end{bmatrix}$

$2K|E|$ scalar messages

Distributed Denoising - Method 1

- Prior: signal is smooth w.r.t the underlying graph structure

Distributed Denoising - Method 1

- Prior: signal is smooth w.r.t the underlying graph structure
- Regularization term: $f^{\mathrm{T}}\mathcal{L}f = \frac{1}{2}\sum_{n\in V}\sum_{m\sim n} w_{m,n}\left[f(m)-f(n)\right]^2$
  - $f^{\mathrm{T}}\mathcal{L}f = 0$ iff f is constant across all vertices
  - $f^{\mathrm{T}}\mathcal{L}f$ is small when signal f has similar values at neighboring vertices connected by an edge with a large weight

Distributed Denoising - Method 1

- Prior: signal is smooth w.r.t the underlying graph structure

- Regularization term: $f^{\mathrm{T}}\mathcal{L}f = \frac{1}{2}\sum_{n\in V}\sum_{m\sim n} w_{m,n}\left[f(m)-f(n)\right]^2$
    - $f^{\mathrm{T}}\mathcal{L}f = 0$ iff f is constant across all vertices
    - $f^{\mathrm{T}}\mathcal{L}f$ is small when signal f has similar values at neighboring vertices connected by an edge with a large weight

- Distributed regularization problem:

$$\underset{f}{\operatorname{argmin}} \frac{\tau}{2}\|f-y\|_2^2 + f^{\mathrm{T}}\mathcal{L}f \tag{1}$$

Distributed Denoising - Method 1

- Prior: signal is smooth w.r.t the underlying graph structure
- Regularization term: $f^{\mathrm{T}}\mathcal{L}f = \frac{1}{2}\sum_{n\in V}\sum_{m\sim n} w_{m,n}\left[f(m)-f(n)\right]^2$
  - $f^{\mathrm{T}}\mathcal{L}f = 0$ iff f is constant across all vertices
  - $f^{\mathrm{T}}\mathcal{L}f$ is small when signal f has similar values at neighboring vertices connected by an edge with a large weight
- Distributed regularization problem:

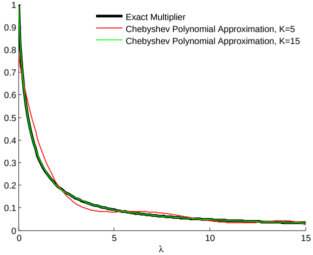
$$\underset{f}{\operatorname{argmin}}\ \frac{\tau}{2}\|f-y\|_2^2 + f^{\mathrm{T}}\mathcal{L}f \tag{1}$$

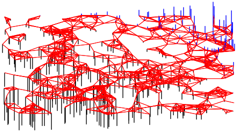
Proposition

The solution to (1) is given by Ry, where R is a graph Fourier multiplier operator with multiplier $g(\lambda_\ell) = \frac{\tau}{\tau+2\lambda_\ell}$.

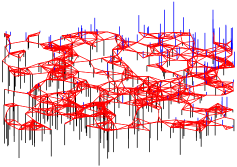
Distributed Denoising Illustrative Example

- Graph analog to low-pass filtering
- Modify the contribution of each Laplacian eigenvector
  - $f_*(n) = (Ry)_n = \sum_{\ell=0}^{N-1} \left[\frac{\tau}{\tau+2\lambda_\ell}\right] \hat{y}(\ell)\chi_\ell(n)$
- Use Chebyshev approximation to compute $\tilde{R}y$ in a distributed manner
- Over 1000 experiments, average mean square error reduced from 0.250 to 0.013

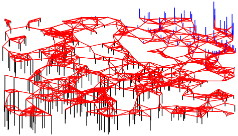




Original Signal



Noisy Signal



Denoised Signal

Distributed Denoising - Method 2

- Prior: signal is p.w. smooth w.r.t. graph ⇔ SGWT coefficients sparse

Distributed Denoising - Method 2

- Prior: signal is p.w. smooth w.r.t. graph ⇔ SGWT coefficients sparse
- Regularize via LASSO (Tibshirani, 1996):

$$\min_{a} \ \frac{1}{2}\|y - W^* a\|_2^2 + \mu\|a\|_1$$

Distributed Denoising - Method 2

- Prior: signal is p.w. smooth w.r.t. graph ⇔ SGWT coefficients sparse
- Regularize via LASSO (Tibshirani, 1996):

$$\min_{a} \ \frac{1}{2}\|y - W^* a\|_2^2 + \mu\|a\|_1$$

- Solve via iterative soft thresholding (Daubechies et al., 2004):

$$a^{(\beta)} = \mathcal{S}_{\mu\tau}\left(a^{(\beta-1)} + \tau W\left(y - W^* a^{(\beta-1)}\right)\right), \ \beta = 1, 2, \ldots$$

Distributed Denoising - Method 2

- Prior: signal is p.w. smooth w.r.t. graph ⇔ SGWT coefficients sparse
- Regularize via LASSO (Tibshirani, 1996):

$$\min_{a} \ \frac{1}{2}\|y - W^* a\|_2^2 + \mu\|a\|_1$$

- Solve via iterative soft thresholding (Daubechies et al., 2004):

$$a^{(\beta)} = \mathcal{S}_{\mu\tau}\left(a^{(\beta-1)} + \tau W\left(y - W^* a^{(\beta-1)}\right)\right),\ \beta = 1, 2, \ldots$$

- D-LASSO (Mateos et al., 2010) solves in distributed fashion, but requires $2|E|$ messages of length $N(J+1)$ at each iteration

Distributed Denoising - Method 2

- Prior: signal is p.w. smooth w.r.t. graph ⇔ SGWT coefficients sparse
- Regularize via LASSO (Tibshirani, 1996):

$$\min_{a} \; \tfrac{1}{2}\|y - W^* a\|_2^2 + \mu\|a\|_1$$

- Solve via iterative soft thresholding (Daubechies et al., 2004):

$$a^{(\beta)} = \mathcal{S}_{\mu\tau}\left(a^{(\beta-1)} + \tau W\left(y - W^* a^{(\beta-1)}\right)\right), \; \beta = 1, 2, \ldots$$

- D-LASSO (Mateos et al., 2010) solves in distributed fashion, but requires $2|E|$ messages of length $N(J+1)$ at each iteration
- We solve the LASSO with the approximate wavelet operator via the distributed Chebyshev computation method

Distributed Denoising - Method 2

- Prior: signal is p.w. smooth w.r.t. graph ⇔ SGWT coefficients sparse
- Regularize via LASSO (Tibshirani, 1996):

$$\min_{a} \; \tfrac{1}{2}\|y - W^* a\|_2^2 + \mu\|a\|_1$$

- Solve via iterative soft thresholding (Daubechies et al., 2004):

$$a^{(\beta)} = \mathcal{S}_{\mu\tau}\left(a^{(\beta-1)} + \tau W \left(y - W^* a^{(\beta-1)}\right)\right), \;\beta = 1, 2, \ldots$$

- D-LASSO (Mateos et al., 2010) solves in distributed fashion, but requires $2|E|$ messages of length $N(J+1)$ at each iteration
- We solve the LASSO with the approximate wavelet operator via the distributed Chebyshev computation method
- The communication workload only scales with network size through $|E|$, otherwise independent of N

Distributed Denoising - Method 2

- Prior: signal is p.w. smooth w.r.t. graph ⇔ SGWT coefficients sparse
- Regularize via LASSO (Tibshirani, 1996):

$$\min_{a} \ \tfrac{1}{2}\|y - W^* a\|_2^2 + \mu\|a\|_1$$

- Solve via iterative soft thresholding (Daubechies et al., 2004):

$$a^{(\beta)} = \mathcal{S}_{\mu\tau}\left(a^{(\beta-1)} + \tau W\left(y - W^* a^{(\beta-1)}\right)\right),\ \beta = 1, 2, \ldots$$

- D-LASSO (Mateos et al., 2010) solves in distributed fashion, but requires $2|E|$ messages of length $N(J+1)$ at each iteration
- We solve the LASSO with the approximate wavelet operator via the distributed Chebyshev computation method
- The communication workload only scales with network size through $|E|$, otherwise independent of N
- $\|\tilde{W}^* \tilde{a}_* - W^* a_*\|_2^2 \leq \frac{\|y\|_2^3}{\mu}\sqrt{N(J+1)}B(K)$

Distributed Deconvolution/Deblurring

- Noisy observation: $y = \Phi x +$ noise, where Φ is a graph Fourier multiplier operator with multiplier g_{Φ}

- Distributed regularization problem:

$$\operatorname*{argmin}_{f} \frac{\tau}{2} \|y - \Phi f\|_2^2 + f^{\mathrm{T}} \mathcal{L}^r f \tag{2}$$

Distributed Deconvolution/Deblurring

- Noisy observation: $y = \Phi x +$ noise, where Φ is a graph Fourier multiplier operator with multiplier g_Φ

- Distributed regularization problem:

$$\operatorname*{argmin}_f \frac{\tau}{2}\|y - \Phi f\|_2^2 + f^{\mathrm{T}} \mathcal{L}^r f \tag{2}$$

Proposition

The solution to (2) is given by Ry, where R is a graph Fourier multiplier operator with multiplier $g(\lambda_\ell) = \frac{\tau g_\Phi(\lambda_\ell)}{\tau g_\Phi^2(\lambda_\ell) + 2\lambda_\ell^r}$.

- Compute $\tilde{R}y$ in a distributed manner

Distributed Semi-Supervised Classification

- Finite number of classes $\{1, 2, \ldots, C\}$
- We know the class labels for l vertices on the graph ($l << N$)
- Want to determine the labels for the other vertices in a distributed manner

Distributed Semi-Supervised Classification

- Finite number of classes $\{1, 2, \ldots, C\}$
- We know the class labels for l vertices on the graph ($l << N$)
- Want to determine the labels for the other vertices in a distributed manner
- Many centralized solutions (e.g., Zhou et al., 2004) force the labels to be smooth with respect to the intrinsic structure of the graph by

$$\underset{j \in \{1,2,\ldots,\kappa\}}{\operatorname{argmax}} \; F^{opt}_{nj}, \text{ where } \mathbf{F}^{opt} \text{ is the solution to}$$

$$\mathbf{F}^{opt} = \underset{\mathbf{F} \in \mathbb{R}^{N \times \kappa}}{\operatorname{argmin}} \sum_{j=1}^{\kappa} \left\{ \tau \|\mathbf{F}_{:,j} - \mathbf{Y}_{:,j}\|_2^2 + \|\mathbf{F}_{:,j}\|_{\mathcal{H}}^2 \right\}$$

$$\mathbf{Y} = \begin{array}{ccc} & \kappa & \\ \begin{array}{c} l \\ \\ \\ u \end{array} & \left(\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ \hdashline 0 & 0 & 0 \\ 0 & 0 & 0 \\ & \vdots & \\ 0 & 0 & 0 \end{array}\right) & N \end{array}$$

$\|\mathbf{f}\|_{\mathcal{H}}^2 = \langle \mathbf{f}, \mathbf{f} \rangle_{\mathcal{H}} := \langle \mathbf{f}, \mathbf{P}\mathbf{f} \rangle = \mathbf{f}^{\mathrm{T}}\mathbf{P}\mathbf{f}$ for different choices of real, symmetric, positive semi-definite matrices $\mathbf{P}$

Distributed Semi-Supervised Classification (cont'd)

- Equivalent to κ separate minimization problems:

$$\mathbf{F}_{:,j}^{opt} = \operatorname*{argmin}_{\mathbf{f} \in \mathbb{R}^N} \left\{ \tau \|\mathbf{f} - \mathbf{Y}_{:,j}\|_2^2 + \mathbf{f}^{\mathrm{T}} \mathbf{P} \mathbf{f} \right\} \tag{3}$$

- Solution to (3) is given by $\mathbf{R}\mathbf{Y}_{:,j}$, where $\mathbf{R}$ is a generalized graph multiplier operator (with respect to $\mathbf{P}$) with a multiplier of $\frac{\tau}{\tau+\lambda}$

- This type of framework provides a way to distribute a number of existing (centralized) semi-supervised classification and regression methods from the machine learning literature

Summary

- A number of distributed signal processing tasks can be represented as applications of graph multiplier operators
- We approximate the graph multipliers by Chebyshev polynomials
- The recurrence relations of the Chebyshev polynomials make the approximate operators readily amenable to distributed computation
- The communication required to perform distributed computations only scales with the size of the network through the number of edges in the communication graph
- The proposed method is well-suited to large-scale networks with sparse communication graphs

Outline

1. Introduction
2. Spectral Graph Theory Background
3. Wavelet Constructions on Graphs
4. Approximate Graph Multiplier Operators
5. Distributed Signal Processing via the Chebyshev Approximation
6. Open Issues and Challenges

Further Reading

Spectral Graph Theory, Laplacian Eigenvectors, and Nodal Domains

F. K. Chung, *Spectral Graph Theory*. Vol. 92 of the CBMS Regional Conference Series in Mathematics, AMS Bokstore, 1997.

T. Bıyıkoğlu, J. Leydold, and P. F. Stadler, *Laplacian Eigenvectors of Graphs*. Lecture Notes in Mathematics, vol. 1915, Springer, 2007.

Spectral Clustering

U. von Luxburg, "A tutorial on spectral clustering," *Stat. Comput.*, vol. 17, no. 4, pp. 395–416, 2007.

Chebyshev Polynomials

J. C. Mason and D. C. Handscomb, *Chebyshev Polynomials*. Chapman and Hall, 2003.

Spectral Graph Wavelet Transform and Distributed Processing

D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Appl. Comput. Harmon. Anal.*, vol. 30, no. 2, pp. 129–150, Mar. 2011.

D. I Shuman, P. Vandergheynst, and P. Frossard, "Chebyshev polynomial approximation for distributed signal processing,"in *Proc. Int. Conf. Distr. Comput. Sensor Sys. (DCOSS)*, Barcelona, Spain, Jun. 2011.

Best Minimax Appoximation

Weierstrass Approximation Theorem

For any continuous function f on $[a, b]$ and any $\epsilon > 0$, there exists a polynomial p such that

$$\|f - p\|_\infty := \sup_{x \in [a,b]} |f(x) - p(x)| < \epsilon.$$

Best Minimax Appoximation

Weierstrass Approximation Theorem

For any continuous function f on $[a, b]$ and any $\epsilon > 0$, there exists a polynomial p such that

$$\|f - p\|_\infty := \sup_{x \in [a,b]} |f(x) - p(x)| < \epsilon.$$

- Catch: The degree of the approximating polynomial may be large
- What is the best we can do when the degree of the approximating polynomial is bounded?
- Consider approximation space $\mathcal{P}_n$, with elements $p_n(x) = a_0 + a_1 x + \ldots + a_n x^n$

Best Minimax Appoximation

Weierstrass Approximation Theorem

For any continuous function f on $[a, b]$ and any $\epsilon > 0$, there exists a polynomial p such that

$$\|f - p\|_\infty := \sup_{x \in [a,b]} |f(x) - p(x)| < \epsilon.$$

- Catch: The degree of the approximating polynomial may be large
- What is the best we can do when the degree of the approximating polynomial is bounded?
- Consider approximation space $\mathcal{P}_n$, with elements $p_n(x) = a_0 + a_1 x + \ldots + a_n x^n$

Questions

1. Does there exist $p_n^* \in \mathcal{P}_n$ such that $\|f - p_n^*\|_\infty = \inf_{p_n \in \mathcal{P}_n} \|f - p_n\|_\infty$?
2. If so, is it unique?
3. What are the characteristic properties of p_n^*?
4. How do we compute p_n^*?

Best Minimax Appoximation

Weierstrass Approximation Theorem

For any continuous function f on $[a, b]$ and any $\epsilon > 0$, there exists a polynomial p such that

$$\|f - p\|_\infty := \sup_{x \in [a,b]} |f(x) - p(x)| < \epsilon.$$

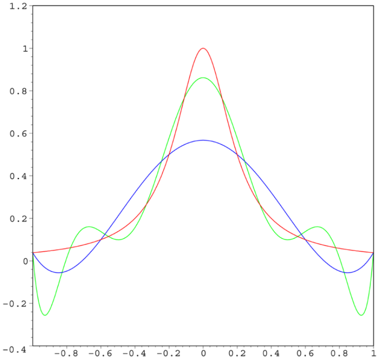
- Catch: The degree of the approximating polynomial may be large
- What is the best we can do when the degree of the approximating polynomial is bounded?
- Consider approximation space $\mathcal{P}_n$, with elements $p_n(x) = a_0 + a_1 x + \ldots + a_n x^n$

Questions

1. Does there exist $p_n^* \in \mathcal{P}_n$ such that $\|f - p_n^*\|_\infty = \inf_{p_n \in \mathcal{P}_n} \|f - p_n\|_\infty$? Yes
2. If so, is it unique? Yes
3. What are the characteristic properties of p_n^*?
4. How do we compute p_n^*?

Polynomial Interpolation and the Runge Phenomenon

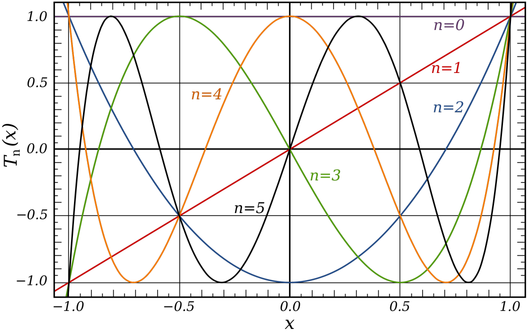
- Fix $n+1$ points in $[-1,1]$
- Unique polynomial of degree n passing through those points
- If you pick $n+1$ points uniformly, max error may increase with n (despite Weierstrass theorem)



Red is function to be approximated, blue is fifth order approx., green is ninth order approx. Source: Wikipedia.

Chebyshev Polynomials

- $T_n(x) := \cos\big(n \arccos(x)\big),\ x \in [-1,1],\ n = 0,1,2,\ldots$
- Chebyshev nodes: $T_n(x) = 0$ at $x_i = \cos\left(\frac{2i-1}{2n}\pi\right)$, $i = 1,2,\ldots,n$
- $T_n(x)$ has $n+1$ extrema at $\cos\left(\frac{k\pi}{n}\right)$, $k = 0,1,\ldots,n$
- Maximum magnitude alternates between 1 and -1 at these $n+1$ points



Source: Wikipedia.

The Minimax Property of Chebyshev Polynomials

Answer to Question 3

- Necessary and sufficient conditions for $\|f - p_n^*\|_\infty = \inf_{p_n \in \mathcal{P}_n} \|f - p_n\|_\infty$

  There exist $n+2$ distinct points $x_1 < x_2 < \ldots < x_{n+2}$ such that:

  - $|f(x_i) - p_n^*(x_i)| = \|f - p_n^*\|_\infty \ , i = 1, 2, \ldots, n+2$
  - Residuals at these points alternate signs

The Minimax Property of Chebyshev Polynomials

Answer to Question 3

- Necessary and sufficient conditions for $\|f - p_n^*\|_\infty = \inf\limits_{p_n \in \mathcal{P}_n} \|f - p_n\|_\infty$

  There exist $n + 2$ distinct points $x_1 < x_2 < \ldots < x_{n+2}$ such that:

  - $|f(x_i) - p_n^*(x_i)| = \|f - p_n^*\|_\infty$, $i = 1, 2, \ldots, n + 2$
  - Residuals at these points alternate signs

Application: $\underset{p_{n-1} \in \mathcal{P}_{n-1}}{\operatorname{argmin}} \|x^n - p_{n-1}\|_\infty = x^n - \frac{1}{2^{n-1}} T_n(x)$

The Minimax Property of Chebyshev Polynomials

Answer to Question 3

- Necessary and sufficient conditions for $\|f - p_n^*\|_\infty = \inf_{p_n \in \mathcal{P}_n} \|f - p_n\|_\infty$

  There exist $n + 2$ distinct points $x_1 < x_2 < \ldots < x_{n+2}$ such that:

  - $|f(x_i) - p_n^*(x_i)| = \|f - p_n^*\|_\infty \ , i = 1, 2, \ldots, n + 2$
  - Residuals at these points alternate signs

Application: $\underset{p_{n-1} \in \mathcal{P}_{n-1}}{\operatorname{argmin}} \|x^n - p_{n-1}\|_\infty = x^n - \frac{1}{2^{n-1}} T_n(x)$

Answer to Question 4

- Polynomial interpolation with the $n + 1$ points chosen to be the Chebyshev nodes (zeros) of $T_{n+1}(x)$
- Puts more of the interpolation points towards the ends than uniform choice
- Can iterate by setting new interpolation points to be those with the largest magnitude of error in previous round
- Near-optimal and the error decreases as you consider higher degree polynomials

Recurrence Relations of Chebyshev Polynomials

1. $T_0(x) = 1$
   $T_1(x) = x$
   $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x) \quad \text{for } k \geq 2$

2. $T_k(x)T_{k'}(x) = \frac{1}{2}\left[T_{k+k'}(x) + T_{|k-k'|}(x)\right]$

Recurrence Relations of Chebyshev Polynomials

1. $T_0(x) = 1$
   $T_1(x) = x$
   $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x) \quad \text{for } k \geq 2$

2. $T_k(x)T_{k'}(x) = \frac{1}{2}\left[T_{k+k'}(x) + T_{|k-k'|}(x)\right]$

Shifted Chebyshev Polynomials

- To shift the domain from [-1,1] to [0,A], define

$$\overline{T}_k(x) := T_k\left(\frac{x}{\alpha} - 1\right), \text{ where } \alpha := \frac{A}{2}$$

- $\overline{T}_k(x) = \frac{2}{\alpha}(x - \alpha)\overline{T}_{k-1}(x) - \overline{T}_{k-2}(x) \quad \text{for } k \geq 2$

Chebyshev Expansion

- Chebyshev polynomials form an orthogonal basis for $L^2\left([-1,1], \frac{dx}{\sqrt{1-x^2}}\right)$

    - $$\langle T_m, T_n \rangle = \int\limits_{-1}^{1} \frac{T_m(x) T_n(x)}{\sqrt{1-x^2}} dx = \begin{cases} 0 & \text{if } m \neq n \\ \frac{\pi}{2} & \text{if } m = n > 0 \\ \pi & \text{if } m = n = 0 \end{cases}$$

    - Every $h \in L^2\left([-1,1], \frac{dx}{\sqrt{1-x^2}}\right)$ can be represented as

$$h(x) = \frac{1}{2}c_0 + \sum_{k=1}^{\infty} c_k T_k(x), \text{ where } c_k = \frac{2}{\pi}\int_0^{\pi} \cos(k\theta) h(\cos(\theta)) d\theta$$

    - Coefficients usually decrease rapidly

- If $h(\cdot)$ has $M+1$ continuous derivatives,

$$\left| h(x) - \left[\frac{1}{2}c_0 + \sum_{k=1}^{K} c_k T_k(x) \right] \right| = \left| \sum_{k=K+1}^{\infty} c_k T_k(x) \right| = \mathcal{O}(K^{-M}), \forall x \in [-1,1]$$