Chebyshev Polynomial Approximation for Distributed Signal Processing

David Shuman, Pierre Vandergheynst, and Pascal Frossard

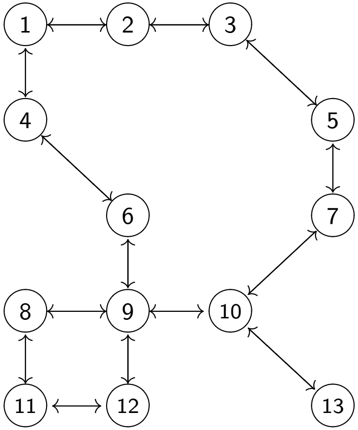
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Signal Processing Laboratory
{david.shuman, pierre.vandergheynst, pascal.frossard}@epfl.ch

International Conference on
Distributed Computing in Sensor Systems (DCOSS)
Barcelona, Spain

June 28, 2011

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



Outline

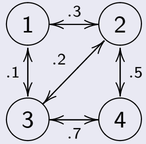
Spectral Graph Theory Notation

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

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

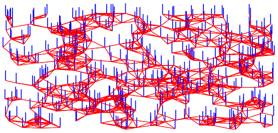


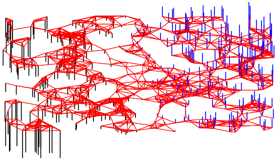
$$W = \begin{bmatrix} 0 & .3 & .1 & 0 \\ .3 & 0 & .2 & .5 \\ .1 & .2 & 0 & .7 \\ 0 & .5 & .7 & 0 \end{bmatrix}$$

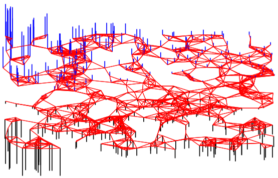
$$D = \begin{bmatrix} .4 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1.2 \end{bmatrix}$$

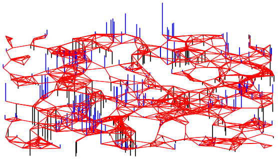
Graph Laplacian Eigenvectors

- 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 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

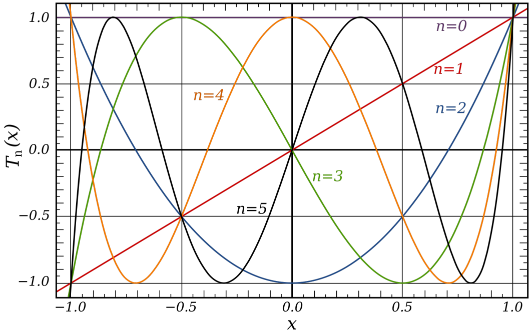
$$\Phi f(n) = \sum_{\ell=0}^{N-1} g(\lambda_\ell)\hat{f}(\ell)\chi_\ell(n)$$

Outline

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

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

- M^{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 M

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

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

- M^{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 M

Shifted Chebyshev Polynomials

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

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

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

Outline

Fast Chebyshev Approx. of a Graph Fourier Multiplier

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

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

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

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

Question: Why do we call this a fast approximation?

Fast Chebyshev Approx. of a Graph Fourier Multiplier

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

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

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

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

Outline

Distributed Computation

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

<u>NODE *n*'S KNOWLEDGE:</u>

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

Distributed Computation

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

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 $c_o, c_1, \ldots, c_M$
4. Loose upper bound on $\lambda_{\max}$

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

Distributed Computation

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

<u>NODE n'S KNOWLEDGE:</u>

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

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

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

Distributed Computation

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

<u>NODE *n*'S KNOWLEDGE:</u>

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

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

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

Distributed Computation

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

<u>NODE n'S KNOWLEDGE:</u>

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

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

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

$2M|E|$ scalar messages

Outline

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

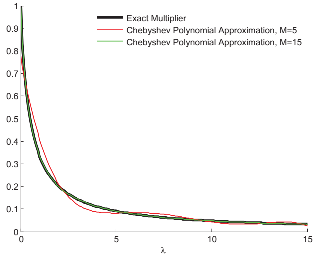
The solution to (1) *is given by* Ry*, where* R *is a graph Fourier multiplier operator with multiplier* $g(\lambda_\ell) = \frac{\tau}{\tau+2\lambda_\ell}$.

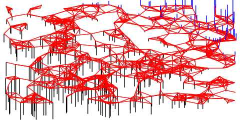
Distributed Denoising Illustrative Example

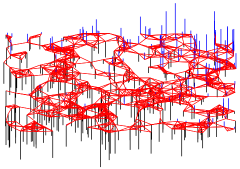
- Graph analog to low-pass filtering
- Modify the contribution of each Laplacian eigenvector

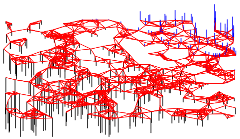
$$f_*(n) = (Ry)_n = \sum_{\ell=0}^{N-1} \left[\frac{\tau}{\tau+2\lambda_\ell}\right] \hat{y}(\ell)\chi_\ell(n)$$

- 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

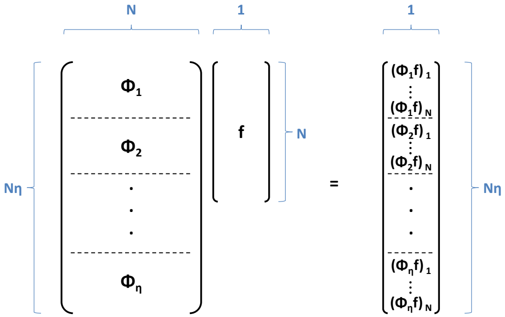
Unions of Graph Fourier Multipliers

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

$$
N\eta\left\{\overbrace{\begin{bmatrix} \Phi_1 \\ \hdashline \Phi_2 \\ \hdashline \cdot \\ \cdot \\ \cdot \\ \hdashline \Phi_\eta \end{bmatrix}}^{N}\right.
\left.\overbrace{\begin{bmatrix} \\ f \\ \\ \end{bmatrix}}^{1}\right\}N
=
\left.\overbrace{\begin{bmatrix} (\Phi_1 f)_1 \\ \vdots \\ (\Phi_1 f)_N \\ \hdashline (\Phi_2 f)_1 \\ \vdots \\ (\Phi_2 f)_N \\ \hdashline \cdot \\ \cdot \\ \cdot \\ \hdashline (\Phi_\eta f)_1 \\ \vdots \\ (\Phi_\eta f)_N \end{bmatrix}}^{1}\right\}N\eta
$$

Unions of Graph Fourier Multipliers

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



EXAMPLE: SPECTRAL GRAPH WAVELET TRANSFORM (HAMMOND ET AL., 2011)

- $(\Phi f)_{(j-1)N+n} = \sum_{\ell=0}^{N-1} g_j(\lambda_\ell)\hat{f}(\ell)\chi_\ell(n)$ for $j \in \{1,2,\ldots,\eta\}$, $n \in \{1,2,\ldots,N\}$
- $g_j(\lambda_\ell) = g(t_j\lambda_\ell)$ for $j \in \{1,2,\ldots,\eta-1\}$, where $g(\cdot)$ is a band-pass filter
- $g_\eta(\cdot)$ is a low-pass filter; coefficients represent low frequency content of signal

Distributed Denoising - Method 2

- Prior: signal is p.w. smooth w.r.t. graph $\Leftrightarrow$ SGWT coefficients sparse

Distributed Denoising - Method 2

- Prior: signal is p.w. smooth w.r.t. graph $\Leftrightarrow$ 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):

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

Distributed Denoising - Method 2

- Prior: signal is p.w. smooth w.r.t. graph $\Leftrightarrow$ 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):

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

- 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 $\Leftrightarrow$ 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):

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

- D-LASSO (Mateos et al., 2010) solves in distributed fashion, but requires $2|E|$ messages of length $N(J+1)$ at each iteration
- Communication cost of Chebyshev polynomial approximation:
  - One computation of $\tilde{W}y$ ($2M|E|$ messages of length 1)
  - At each soft thresholding iteration, distributed computation of $\tilde{W}\tilde{W}^* a^{(k-1)}$ ($2M|E|$ messages of length $J+1$ and $2M|E|$ messages of length 1)
  - One final computation of $\tilde{W}^*\tilde{\tilde{a}}$ to recover signal ($2M|E|$ messages of length $J+1$)

Distributed Denoising - Method 2

- Prior: signal is p.w. smooth w.r.t. graph $\Leftrightarrow$ 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):

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

- D-LASSO (Mateos et al., 2010) solves in distributed fashion, but requires $2|E|$ messages of length $N(J+1)$ at each iteration
- Communication cost of Chebyshev polynomial approximation:
  - One computation of $\tilde{W}y$ ($2M|E|$ messages of length 1)
  - At each soft thresholding iteration, distributed computation of $\tilde{W}\tilde{W}^* a^{(k-1)}$ ($2M|E|$ messages of length $J+1$ and $2M|E|$ messages of length 1)
  - One final computation of $\tilde{W}^*\tilde{\tilde{a}}$ to recover signal ($2M|E|$ messages of length $J+1$)
- **Key takeaway: communication workload only scales with network size through $|E|$, otherwise independent of N**

Outline

Summary

- Graph Fourier muliplier operators are the graph analog of filter banks
  - They reshape functions' frequencies through multiplication in the graph Fourier domain
- A number of distributed signal processing tasks can be represented as applications of graph Fourier multiplier operators
- We approximate graph Fourier 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 sensor networks with sparse communication graphs

Ongoing Work and Extensions

- Reviewer: *"This seems to be a very interesting technique looking for a problem"*
    - Other possible applications we are working on include distributed smoothing, deconvolution, classification, and learning
    - More thorough comparisons of communication costs with alternative distributed methods for these applications

Ongoing Work and Extensions

- Reviewer: *"This seems to be a very interesting technique looking for a problem"*
  - Other possible applications we are working on include distributed smoothing, deconvolution, classification, and learning
  - More thorough comparisons of communication costs with alternative distributed methods for these applications
- Extension: use the eigenvectors of other symmetric positive-semidefinite matrices as bases

Ongoing Work and Extensions

- Reviewer: *"This seems to be a very interesting technique looking for a problem"*
  - Other possible applications we are working on include distributed smoothing, deconvolution, classification, and learning
  - More thorough comparisons of communication costs with alternative distributed methods for these applications
- Extension: use the eigenvectors of other symmetric positive-semidefinite matrices as bases
- Robustness issues
  - Sensitivity to quantization and communication noise - how do they propagate?
  - Effect of a sensor node dropping out of the network or losing synchronicity

Further Reading

Spectral Graph Theory and Laplacian Eigenvectors

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.

Chebyshev Polynomials

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

G. M. Phillips, *Interpolation and Approximation by Polynomials*. CMS Books in Mathematics, Springer-Verlag, 2003.

T. J. Rivlin, *Chebyshev Polynomials*. Wiley-Interscience, 1990.

Spectral Graph Wavelet Transform

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.