WAVELETS ON GRAPHS

SHASHANK SULE

Contents

Laplacian + Operator theory = Wavelets 1
To and from the wavelet domain 3
Questions 6
References 6

Laplacian + Operator theory = Wavelets

OK, in this note we talk about wavelets on graphs. Note that we work with the non-normalized graph Laplacian:

$$L = V\Lambda V^*$$

Now, given ANY complete matrix T we can define a function on T by:

$$g(T) := Qg(\Lambda)Q^{-1}$$

where

$$g(\Lambda) = \begin{bmatrix} g(\lambda_1) & & \\ & g(\lambda_2) & \\ & & \ddots \end{bmatrix}$$

We do the same for the graph wavelet kernel g:

$$T_g = Vg(\Lambda)V^*$$

The scaling is done using $T_g = Vg(t\Lambda)V^*$. Finally, the wavelet localized at the vertex n with scaling t is computed using the action of T_g on δ_n, the vector taking the value 1 at the vertex n and 0 otherwise:

$$\psi_{t,n} = T_g\delta_n = Vg(\Lambda)V^*\delta_n$$

Date: November 2019.

The wavelet coefficients can be found by computing the inner product of a given function f with the wavelet $\psi_{t,n}$:

$$W_f(t,n) = \langle \psi_{t,n}, f \rangle$$

Note that when the (real valued) function g is in $L^2_\mu(\mathbb{R})$ where $d\mu = dt/t$ then we can actually recover f from the wavelet coefficients. First, for notational clarity, let Ψ_t denote the $n \times n$ matrix whose jth column is $\psi_{t,j}$ (thus $\Psi_t = Vg(t\Lambda)V^*I$). And the column vector of wavelet coefficients of f is denoted w. Note that $w = \Psi_t^* f$. Furthermore, let

$$C_g = \int_{\mathbb{R}^+} \frac{(g(t))^2}{t}\, dt < \infty$$

Then, look at the following computation:

$$\begin{aligned}
\frac{1}{C_g}\int_{\mathbb{R}^+} \Psi_t w \,\frac{dt}{t} &= \frac{1}{C_g}\int_{\mathbb{R}^+} \Psi_t \Psi_t^* f \,\frac{dt}{t} \\
&= \frac{1}{C_g}\int_{\mathbb{R}^+} Vg(t\Lambda)V^*IVg(t\Lambda)V^* f \,\frac{dt}{t} \\
&= \frac{1}{C_g}\int_{\mathbb{R}^+} V(g(t\Lambda))^2 V^* f \,\frac{dt}{t} \\
&= V\frac{1}{C_g}\int_{\mathbb{R}^+} \frac{g(t\Lambda)}{t}\, dt V^* f \\
&= V\begin{bmatrix} 0 & \\ & I_{n-1} \end{bmatrix} V^* f \\
&= f - \hat{f}(0)\chi_0
\end{aligned}$$

Here χ_n is the nth Fourier basis vector. This is a more notation friendly version of Lemma 1 from [1].

The rest of the lemmas are pretty technical. Lemmas 5.2,3,and 4 are graph theoretic and Lemma 5.5 says that if two kernel functions are within a given bound then the respective entries of their corresponding wavelets are also within the same bound. Lemma 5.6 says that given a kernel which is $K+1$ times continuously differentiable, it is within a $K+1$ degree polynomial (in t) of a kernel which is a K degree polynomial. Finally, theorem 5.7 states that the value taken at vertex m by a normalized wavelet localized at vertex n grows at most linearly with t when t is sufficiently small and the prefactor is given by the regularity. So it actually makes sense to not have too-regular a kernel, because the prefactor might get too large, if the maximal eigenvalue is bigger than 1. If the maximal eigenvalue is less than 1, then use as large a regularity as possible!

On scaling functions: Wavelets defined via the wavelet operator might not pick up low frequencies so we need a "lowpass" filter, given by another kernel h which has different properties at 0, i.e $h(0) \neq 0$.

The next main feature is that we approximate the Wavelet operator via a polynomial, which magically eases the computation and recovery in the Wavelet frame.

To and from the wavelet domain

First observe that we can only compute the Wavelet coefficients for finitely many scales. So we tweak our notation a little bit and say that $W_f(t_j, n)$ is the wavelet coefficient for f scaled at t_j localized at n. Suppose there are J scales and N vertices. Counting the low-pass coefficients (with scale equal to 1) we get $N(J+1)$ coefficients:

$$w_j = \Psi^*_{t_j} f$$

Also observe that Ψ_t is self-adjoint so $w_j = \Psi_{t_j} f$.

Now, computing these coefficients is hard because we need to do a spectral decomposition on the laplacian and that is a truly unstable problem when the graph is dense. Instead, we approximate Ψ_{t_j} cleverly with Chebyshev polynomials. This is motivated due to the fact that

$$\Psi_{t_j} = V g(t_j \Lambda) V^* I = g(t_j L)$$

But g has an expansion in the basis of Chebyshev polynomials. Although Chebyshev polynomials are canonically defined on the interval $[-1, 1]$ they may be supplanted to any closed interval via a linear transformation. In particular, since we only really care about the values g takes from 0 to $t_j \lambda_{\max}$, we transplant the Chebyshev polynomials to $[0, t_j \lambda_{\max}]$ for each scale. For notational convenience, we just denote the nth Chebyshev polynomial for the jth scaling factor as T^j_n.

$$g(t_j x) = \sum_{n=0}^{\infty} c_{j,n} T^j_n(x)$$

We approximate $g(t_j x)$ by using only the first M_j terms:

$$g(t_j x) \approx p_j(x) = \sum_{n=0}^{M_j} c_{j,n} T^j_n(x)$$

The corresponding matrix equation is

$$g(t_jL) \approx p_j(L) = \sum_{n=0}^{M_j} c_{j,n} T_n^j(L)$$

Then

$$p_j(L)f = \sum_{n=0}^{M_j} c_{j,n} T_n^j(L)f$$

This is where the Chebyshev magic comes in: we exploit the recurrence relation

$$T_{k+2}^j(L) = q(L)T_{k+1}^j(L) - T_k^j(L)$$

Here q is just a linear polynomial that comes out of adjusting the Chebyshev recurrence relation for the supplanting. Basically the above identity tells us that we only need to compute the first two evaluations of the polynomial for every scale; the rest can be found via the recurrence relation. Of course, the first two Chebyshev polynomials are the 1 polynomial and a linear polynomial so we needn't spectrally decompose the Laplacian. So we have a Fast Wavelet transform, W, one that takes a f from $\mathbb{R}^N = \mathbb{R}^{|V|}$ to $\mathbb{R}^{(J+1)N}$. The action of W on f is given as follows:

$$Wf = \begin{bmatrix} p_0(L)f \\ \vdots \\ p_J(L)f \end{bmatrix}$$

So we get the wavelet coefficients with relative ease. What about reconstruction? Observe that $Wf = w$ is an overcomplete system because we have too many equations. So we will go for the least squares approximation to f:

$$f = \underset{x}{\operatorname{argmin}} ||Wx - w||_2$$

From some least squares approximation theory found in [2] we know that

$$f = (W^*W)^{-1}W^*w$$

So we need to compute the adjoint of the Wavelet transform, W^* (and hopefully be able to invert it). Recall that when $T : H_1 \mapsto H_2$ is a linear map between two Hilbert spaces, the (Hermitian) adjoint of T denoted T^* is the map from H_2 to H_1 such that

$$\langle Tf, g\rangle_{H_2} = \langle f, T^* g\rangle_{H_1}$$

For finite dimensional vector spaces the Hermitian adjoint coincides with the adjoint of the map so it can be computed as the conjugate transpose of a matrix. What is the adjoint of the Wavelet operator? Note that when $g \in \mathbb{R}^{N(J+1)}$ then we can express it via the block form:

$$g = \begin{bmatrix} g_0 \\ \vdots \\ g_J \end{bmatrix}$$

We can use the block form to compute the inner product, viewing g and Wf as $J+1$-dimensional vectors where the elements are themselves N dimensional:

$$\begin{aligned} \langle Wf, g\rangle_{\mathbb{R}^{N(J+1)}} &= \sum_{j=0}^{J} \langle p_j(L)f, g_j\rangle_{\mathbb{R}^N} \\ &= \sum_{j=0}^{J} \langle f, p_j(L)g_j\rangle_{\mathbb{R}^N} \\ &= \langle f, \sum_{j=0}^{j} p_j(L)g_j\rangle_{\mathbb{R}^N} \\ &= \langle f, W^* g\rangle \end{aligned}$$

So the operator W^* that takes g to $\sum_{j=0}^{j} p_j(L)g_j$ is the adjoint we were looking for. Furthermore, it can also be computed fast due to the fast polynomial computation.

But, W^*W can be computed even faster. Look at the following calculation:

$$W^*W = \sum_{j=0}^{J} p_j(L)(p_j(L)f) = \Big(\sum_{j=0}^{J} p_j(L)\Big)^2 f$$

Set $P(x) = (\sum_{j=0}^{J} p_j(x))^2$ then the operator in the above equation is just $P(L)$. Finally, we try to represent $P(x)$ in the Chebyshev basis. But we run into a problem because each p_j is a linear combination of Chebyshev polynomials over different intervals. For example, if $t_1 = 1$ and $t_2 = 2$ then p_1 is a linear combination of Chebyshevs over $[0, \lambda_{\max}]$ while p_2 is a linear combination of Chebyshevs over $[0, 2\lambda_{\max}]$. The cheap and dirty way is to pick the maximal t_j so that all possible

$t_j\lambda$ are in the interval $[0, t_{\max}\lambda_{\max}]$. So we can use just one system of Chebyshev polynomials for all scales (we'll just call them T_n, dropping the subscript). Now we are ready to get the expansion for $P(x) = \sum_{k=0}^{2\max_j M_j} d_k T_k$. The coefficients can be recovered pretty easily due to the properties of the Chebyshev polynomials. Lastly, we find a formula for W^*Wf:

$$W^*Wf = P(L)f = \sum_{k=0}^{2\max_j M_j} d_k T_L f$$

Once again, we can use the recurrence relation on the Chebyshev polynomials to avoid the spectral calamity of the Laplacian.

Conclusion: We have W^* and W^*W. Invert the latter and multiply to recover.

Questions

(1) Is there an error-optimal set of scales t_j or approximation degree M_j? What are some possible designs?

(2) Can we use orthogonal polynomials other than Chebyshevs? Seemingly the most attractive property is not the recurrence relation but the multiplication formula. Furthermore, they are the sup norm maximizing polynomials.

(3) Find non-polynomial kernel and low-pass designs.

(4) I feel cheated. Eventually, we have to invert a large system which isn't great, even though the matrix is symmetric because a bad condition number spells the end for Cholesky (or for gradient descent). What about inversion-friendly designs? What can we even say about the condition number for W^*W? Same goes for diffusion maps.

References

[1] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. "Wavelets on graphs via spectral graph theory". In: *Applied and Computational Harmonic Analysis* 30.2 (2011), pp. 129–150.

[2] Peter J Olver, Chehrzad Shakiban, and Chehrzad Shakiban. *Applied linear algebra*. Vol. 1. Springer, 2006.