ELEC-573 Project Final Report

Negar Erfanian
Department of Electrical Engineering
Rice University
Houston, TX 77005
ne12@rice.edu

Abstract

The application of scattering transforms, as non-trainable deep convolutional architects, to network data has been of interest in recent years. These transforms are composed of two or more layers of wavelet filterbanks in such a way that extract inherent structures of data while preserving the information. Moreover, they are proven to be translation invariant and stable to small deformations of the original data. This project aims to study the extension of scattering transforms into graphs as non-euclidean spaces and then use them as a classification method on network data.

1 Introduction

In recent years, graphs have been of significant interest in studying the geometric structures of data in different applications such as classification. The underlying connection between many datasets can be analyzed using graphical signal processing. Therefore, applying a classification tool to these non-euclidean data structures can be tricky. The reason for that is when studying graphs, many signal processing techniques such as filtering, translation, and convolution are not defined the same way as processing data in a Euclidean domain [1]. Mallat et al. introduced a novel classification technique that relies on cascading wavelet transforms on time-series and needs no training on the available dataset [2]. This tool, called the scattering transform, is shown to be invariant to translation, stable to small deformations, and energy preservable that are the critical points of a reasonable classification method. More on this technique is the non-necessity of having a large dataset, making it an ideal classifier. Recently, [3] showed that an extension of a scattering transform to graphical models also have the three mentioned characteristics: time-invariant, stable to perturbations, and energy-preservable. Therefore, in this project, I study scattering transforms' application into graphs as a non-trainable classification technique.

The remainder of this report is organized as follows. I first give a short introduction to the structure of a scattering transform. Next, I discuss the different proposed graph wavelets and present the one I chose to work with for this project. I then present the extension of the scattering transform to graphs using the introduced graph wavelets. Next, data collected for this work will be introduced. Finally, I present how the classification results using the linear SVM on graph scattering coefficients outperforms those of the same classifier built upon the graph Fourier coefficients, while using the same network data.

2 Scattering Transform

Scattering Transform was first introduced in the work of [5] while presenting a classifier using time-series. Given a signal $x(t)$, we look for a nonlinear presentation $\Phi(x)$ that has three main characteristics as

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

- Invariant to translation: $\Phi(x(t-c)) = \Phi(x(t))$ where c is a constant.
- Stable to small deformations $\|\Phi(x(t-\tau(t))) - \Phi(x(t))\| \leq C\|x\|sup|\nabla\tau(t)|$
- Preserves the energy of a signal: $\|\Phi(x)\|^2 = \|x\|^2$

Mallat et al. in [1] and [5] proved that cascading of a wavelet family on the signal $x(t)$ provides $\Phi(x)$ as a nonlinear transform of a data that has all these characteristics and is, therefore, an excellent candidate for classification.

Having a family of dilated wavelets $\psi_\lambda(t) = 2^{-jQ}\psi(2^{-jQ}t)$ where $\lambda = 2^{-jQ}$ and a low-pass filter $\phi(t)$ we can write the wavelet transform of a signal $x(t)$ as

$$Wx(t) = \{x * \phi(t),\ \ x * \psi_\lambda(t)\}_\lambda, \tag{1}$$

where $*$ accounts for convolution. Therefore, for any path $p = (\lambda_1, \lambda_2, \lambda_3, \cdots, \lambda_m)$ of order m, we can write our scattering coefficients as

$$S[p]x(t) = ||\cdots||x * \psi_{\lambda_1}(t)| * \psi_{\lambda_2}(t)| * \cdots| * \psi_{\lambda_m}(t)| * \phi(t). \tag{2}$$

The absolute value $|x*\psi_\lambda(t)|$ in (2) pushes the higher frequency contents towards the low frequencies that can be finally captured by the low-pass filter $\phi(t)$. Moreover, cascading the wavelet transform in (2) prevents loss of information in the high frequencies. Therefore, our final scattering coefficients $S[p]x(t)$ that we earlier showed by $\Phi(x)$ are the ideal nonlinear transformed data on which can do classification. We can use these coefficients as features of data samples to use a linear classifier with labeled data.

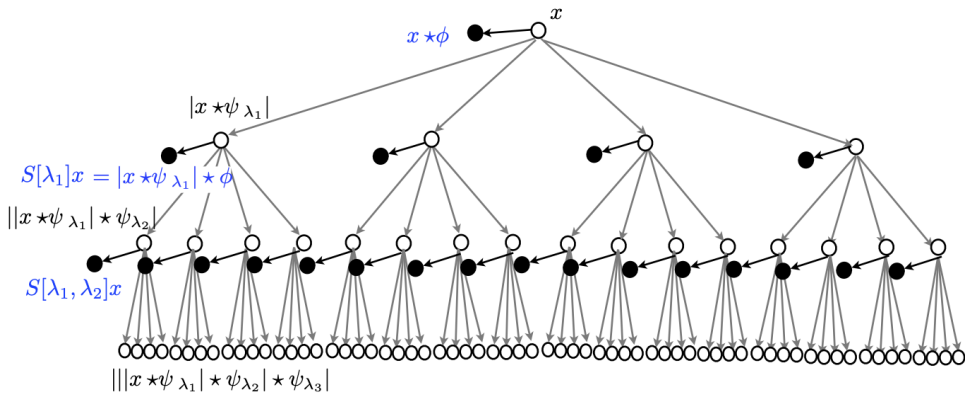


Figure 1: Representation of the scattering transform of a time-series data $x(t)$. The 0th layer's output accounts for the signal mean, whereas the outputs on the first and second layers are the mean of the modulus of the wavelet transforms. Only the filled dark circles are the outputs, and the inner wavelet coefficients at the very last layer are very close to zero. This is because almost all the signal's energy has been captured by the 0th, first, and second layer outputs, and therefore having two layers is efficient.

Figure 1 represents the structure of a scattering network. Unlike the neural networks that outputs are in the final layer, the scattering network outputs are in each layer, as shown by the black filled circles. As depicted in this figure, the 0th layer represents the average of signal $x(t)$. To choose the number of layers to cascade the wavelet transforms, [6] has proved the efficiency of $m = 2$ layers to preserve all the energy.

Figure 2 shows a family of Morlet wavelets in the frequency domain that refers to the wavelet family introduced in (1).

As explained earlier, the scattering transform introduced in (2) can be extended to the network data represented by graphs. Before we get into that part, we first provide a summary of graph wavelets that we need to perform cascading on in the next section.

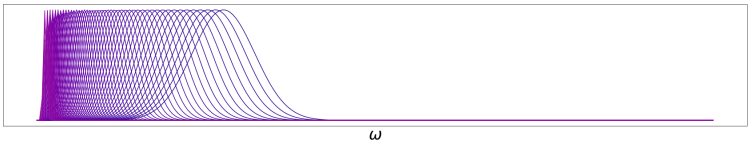


Figure 2: Dilated Morlet wavelets in the frequency domain.

3 Graph Wavelets

If we represent our network data with graph $G = (V, E, W)$ where V represents the set of N nodes, $E : V \times V$ accounts for the existing edges, and W represents the edge weights, the combinatorial graph Laplacian is denoted by $L = D - W$ where D is the degree matrix. Therefore, as represented by the work of [1], the laplacian eigendecomposition represents the Fourier spectrum of graph G as shown by

$$LU_G = \Lambda U_G, \tag{3}$$

where U_G and Λ represent the eigenvectors and eigenvalues of the combinatorial graph Laplacian L. Therefore having any signal $x[n]$, $\{n \in 0, 1, \cdots, N-1\}$ that is translated to graph G, we can write the fourier transorm of $x[n]$ as

$$\hat{x} = U_G^H x, \tag{4}$$

where $\hat{x}$ represents the Fourier transform of x, and ^{H} stands for the hermitian matrix. The reason behind introducing the graph Fourier is that dealing with graph Fourier multiplication is much easier than graph convolution when introducing the graph wavelet transform. As we know from signal processing, the multiplication in the Fourier domain is the same as a convolution in the time domain. Therefore, here we present the graph wavelets in the Fourier domain.

To perform wavelet transform of a signal on a graph, we need to look for kernel $h(\lambda)$ that captures the frequency spectrum of graph G that is represented by its Laplace eigenvalues λ as shown earlier in (3). To my knowledge, four different kernels have been introduced so far in the work of [3-4,7-9]. After exploring all the proposed kernels, I decided to implement the spectrum-adapted tight graph wavelet based on Hann wavelets as represented in [7], due to forming tight frames and being localized in both vertex and frequency domain. A representation of these wavelets using the Minnesota graph model is shown in Fig. 3.

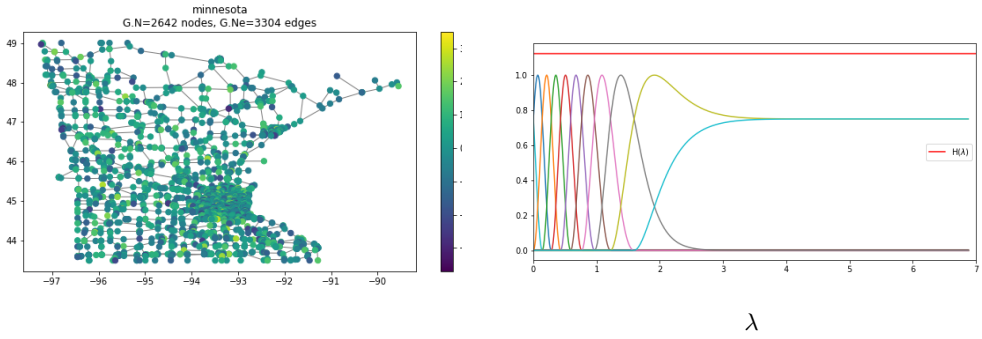


Figure 3: Representation of the spectrum-adapted tight graph wavelet kernels using hann kernels. The left figure shows the Minnesota graph model with random signals being assined to its nodes. The figure on right shows the graph wavelet kernels that are desined to be adapted to the laplacian eigenvalues spectrum of the Minnesota graph. The red horizontal line named by $H(\lambda)$ proves that these kernels form a tight frame. More details on designing these kernels can be found in the Appendix.

After implementing graph kernel $h(\lambda)$, the wavelet transform of signal $x[n]$ can be written as

$$\hat{y}(\lambda) = h(\lambda)\hat{x}(\lambda)$$

$$U_G^H y[n] = h(\lambda) U_G^H x[n]$$
$$y[n] = U_G h(\lambda) U_G^H x[n]$$
$$y[n] = h(L)x[n]. \tag{5}$$

Implementation details about h(L) can be found in the Appendix.

4 Graph Scattering Transform

After introducing and implementing the graph wavelet kernels and taking the graph wavelet transform of data $x[n]$ that is located on graph G, we can combine what we presented in sections 2 and 3 to execute the graph scattering transform. Therefore, instead of having signal $x(t)$ in section 2, we use the graph signal $x[n]$, and instead of using wavelet coefficients introduced in (1), we use the graph wavelet coefficients shown by $y[n]$ in (5). We thus can rewrite (2) as

$$S[p]x[n] = \frac{1}{N}\sum_{n=1}^{N} |h_m[L]| \cdots |h_2[L]|h_1[L]x[n]|| \cdots ||. \tag{6}$$

As shown earlier in 2, we can conclude that using $m = 2$ layers we can capture all the preserved energy in signal $x[n]$. Therefore, we build our network based on 2 layers of cascading the graph wavelet transforms. We also have to mention that as we had in section 2, we always have a 0 layer that accounts for the mean of the absolute value of signal $x[n]$. Figure 4 shows the graph scattering network structure, where black filled circles in each layer show the outputs.

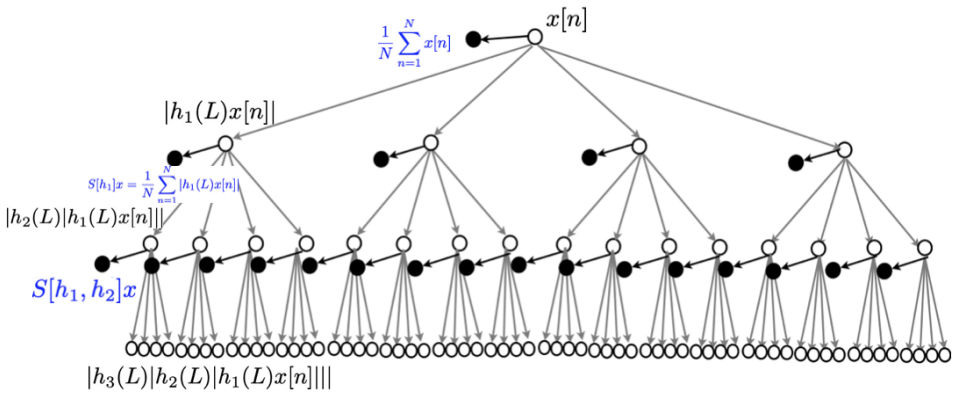


Figure 4: Representation of the scattering transform of a network data $x[n]$. The 0th layer's output accounts for the signal being averaged over the nodes, whereas the outputs on the first and second layers are the mean of the modulus of the graph wavelet transforms. the outputs are represented by the dark circles, and the last layer merges to zero as signal's has been captured and output by the 0th, first, and second layer.

5 Data Collection

In this section, I describe how I collected the network data for this project. I assigned 4 different network data to the graph of Minnesota representing 4 different classes. Class 1 represents random noise signal, whereas classes 2, 3, and 4, are formed using band-limited signal, summation of sinus signal with noise, and summation of sinus signal without noise, respectively. I collected 100 samples for each class, which resulted in 400 network data samples. Listing 1 represents the python code regarding data collection from the 4 mentioned classes. After collecting the data, I extracted the graph scattering and Fourier coefficients for each data sample to which a label is assigned. In the next section, I show how the graph scattering coefficients result in distinguishable clusters.

Listing 1: Data Collection

```
X1 = np.empty((100,len(G.e)))
X2 = np.empty((100,len(G.e)))
X3 = np.empty((100,len(G.e)))
X4 = np.empty((100,len(G.e)))

for i in range(100):
    X1[i,:] = np.random.randn(len(G.e))
    X2[i,:] = band_limited_noise(1, 10, samples=len(G.e), samplerate=10)
    X3[i,:] = np.sin(np.array(range(len(G.e)))/0.7)*np.random.random(1)
        + np.sin(np.array(range(len(G.e)))/1.1)*np.random.random(1)
        + np.sin(np.array(range(len(G.e)))/1.5)*np.random.random(1)
    X4[i,:] = np.sin(np.array(range(len(G.e)))/0.7)
        + np.sin(np.array(range(len(G.e)))/1.1)
        + np.sin(np.array(range(len(G.e)))/1.5)
```

6 t-SNE visualization

t-SNE is a nonlinear dimensionality reduction technique well-suited for embedding high-dimensional data for visualization in a low-dimensional space of two or three dimensions. Figure 5, shows the application of t-SNE on the scattering and Fourier coefficients extracted from all 400 signal samples using two dimensions. As shown in this figure, all 4 classes are linearly and vividly separable when we rely on the graph scattering coefficients of these signal, whereas, these classes are not easily distinguishable using the graph Fourier coefficients. In the next section, we use both scattering and Fourier coefficents of data samples from classes 1 and 2 to apply linear SVM for binary classification.

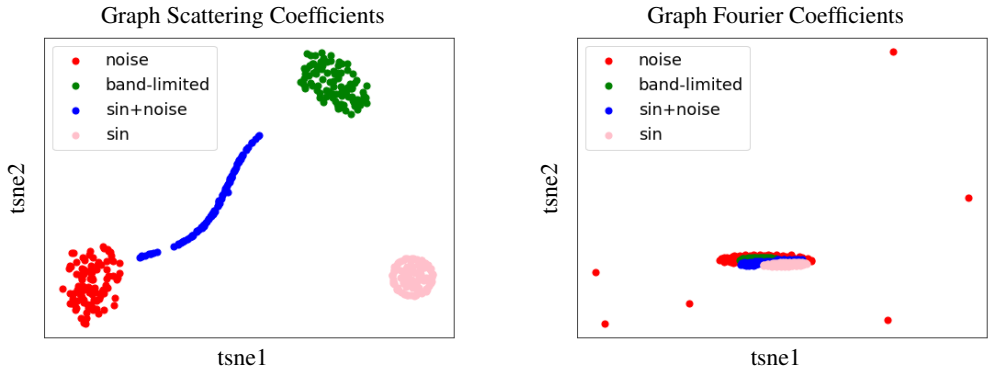


Figure 5: Application of t-SNE technique to the scattering and Fourier coefficients extracted from 4 different classes of graph network data. The left figure represents distinguishable clusters of the 4 classes when we use the graph scattering coefficients. The right figure shows that the 4 classes are not easily separable relying on the graph Fourier coefficients.

7 Classification Results

In this section, we applied the linear SVM and SVM with rbf kernel to the coefficients extracted from graph scattering and Fourier transforms using labeled data of classes 1 and 2. We used the universally accepted sensitivity and specificity metrics to evaluate the performance of our classification method. Table 1 shows the classification results based on the graph scattering and Fourier coefficients. As shown in this table, using a linear classifier, the coefficients extracted using graph scattering transform provide distinguishable features of the two classes, whereas, the graph fourier coefficients do not. Therefore, we can linearly classify the two classes based on the scattering coefficients. On the other hand, we see both methods perform similarly when we use the nonlinear SVM using rbf kernel. This

results show that the coefficients extracted using both graph scattering and fourier transforms provide distinguishable information about the two classes, but are not linearly separable.

Method	Classification	Sensitivity (%)	Specificity (%)	Accuracy (%)
GST	linear SVM	100	100	100
GFT	linear SVM	100	0	43
GST	kernel SVM	100	100	100
GFT	kernel SVM	100	100	100

Table 1: Results obtained using linear and nonlinear SVM using both graph scattering and Fourier coefficients; GST - Graph Scattering Transform, GFT - Graph Fourier Transform.

8 Discussions and Conclusions

In this project, I studied the graph scattering transforms as the cascades of the graph wavelet transform. I first discussed that while the graph wavelet kernels can be constructed using different strategies, I used the spectrum-adapted tight Hann kernels. My choice was forming tight frames and being adapted to the frequency spectrum of the underlying graph's laplacian. Next, I collected 4 types of data, 100 samples per class, assigned to the Minnesota road graph. Using t-SNE, I showed how the four classes are linearly separable using the graph scattering coefficients, whereas the graph Fourier coefficients did not reveal this separability. Finally, I used both linear and nonlinear SVM to do a binary classification using data samples from the first 2 classes. The classification results showed that the graph scattering coefficients could provide distinguishable information for the two classes that resulted in a perfectly linear classification. On the other hand, the graph Fourier coefficients failed to provide information that can linearly distinguish the two classes. Moreover, the SVM with the RBF kernel showed the same excellent performance in distinguishing the two classes using both type graph coefficients.

As we know, the wavelet transformation causes low-frequency resolution in high frequencies. The idea behind the scattering transform is to push the high-frequency content of the data towards the low frequencies to extract them with high-frequency resolution. This then causes being time-invariant and stable to small deformations in the data. On the other hand, the graph Fourier transforms are not stable to the deformations in the signal. Therefore, the difference between the classification results using these two methods will be significant when the signal samples might might have from graph deformations. We realized that linear classification revealed the graph scattering transform's out-performance transform to classify noise and band-limited signals. In the future, I plan to work with more sophisticated and closer to real-world data with network deformations to compare the performance of the two mentioned methods and graph neural networks. One example of such data can be the weather condition in two different semesters, like summer and winter in the USA, where the underlying structure including population and traffic between cities cause deformation of the underlying graph. I also want to include that, even though I did not use the graph neural networks for this work, I believe it will have the same excellent performance as the graph scattering transform. The only superiority of the graph scattering transform is that it needs no training on the data to learn the underlying function. While the graph scattering coefficients can extract all the information from network data, they perform as good as the graph neural network while being computationally less expensive.

References

[1] Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., & Vandergheynst, P. (2013). The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3), 83-98.

[2] Bruna, J., & Mallat, S. (2013). Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8), 1872-1886.

[3] Gama, F., Ribeiro, A., & Bruna, J. (2018). Diffusion scattering transforms on graphs. *arXiv preprint arXiv*:1806.08829.

[4] Gama, F., Ribeiro, A., & Bruna, J. (2019). Stability of graph scattering transforms. In *Advances in Neural Information Processing Systems* (pp. 8038-8048).

[5] Mallat, S. (2012). Group invariant scattering. Communications on Pure and Applied Mathematics, 65(10), 1331-1398.

[6] Waldspurger, I. (2017, July). Exponential decay of scattering coefficients. In 2017 international conference on sampling theory and applications (SampTA) (pp. 143-146). IEEE.

[7] Shuman, D. I., Wiesmeyr, C., Holighaus, N., Vandergheynst, P. (2015). Spectrum-adapted tight graph wavelet and vertex-frequency frames. IEEE Transactions on Signal Processing, 63(16), 4223-4235.

[8] Hammond, D. K., Vandergheynst, P., Gribonval, R. (2019). The spectral graph wavelet transform: Fundamental theory and fast computation. In Vertex-Frequency Analysis of Graph Signals (pp. 141-175). Springer, Cham.

[9] Sandryhaila, A., Moura, J. M. (2013). Discrete signal processing on graphs. IEEE transactions on signal processing, 61(7), 1644-1656.

Appendix

This section gives more details about the spectrum-adapted tight graph wavelet kernel that I used for this work. This wavelet kernel is based on the work presented in [7]. To have wavelet kernels that are both localized in vertices and the graph Laplacian spectrum, we use a warping function that approximates the cumulative spectral density function of the graph Laplacian eigenvalues. Moreover, we want these family kernels to form a tight frame as

$$H(\lambda) := \sum_{m=1}^{M} [\hat{h}_m(\lambda)]^2. \tag{7}$$

If $H(\lambda)$ is a constant on all the graph Laplacian eigenvalues, then $D := \{h_{i,m}\}_{i:1,\cdots,N;m:1,\cdots,M}$ forms a tight frame, where , N and M are respectively the number of graph nodes and wavelet scales. Based on [7], if we set

$$\hat{h}^U(\lambda) := \sum_{k=0}^{K} a_k [cos(2\pi k(\frac{M+1-R}{R\gamma}\lambda + 0.5)).\mathbb{1}_{\{-\frac{R\gamma}{M+1-R} \leq \lambda < 0\}}], \tag{8}$$

where $K = 1, a_0 = a_1 = 0.5, \gamma = \lambda_{max}$, M is the desired number of filters and R controls the overlap of the shifted kernels, we can realize that $H(\lambda)$ in (7) is a constant if

$$\hat{h}^U_m(\lambda) = \hat{h}^U(\lambda - m\frac{\gamma}{M+1-R}). \tag{9}$$

Moreover, having the warping function $\omega(\lambda)$ that approximates the cumulative spectral density function of all λs, we can define our kernels as

$$\hat{h}_m(\lambda) := \hat{h}^U_{m-1}(\omega(\lambda)), \;\; m = 2, 3, \cdots, M, \tag{10}$$

as the $M-1$ wavelet kernels and

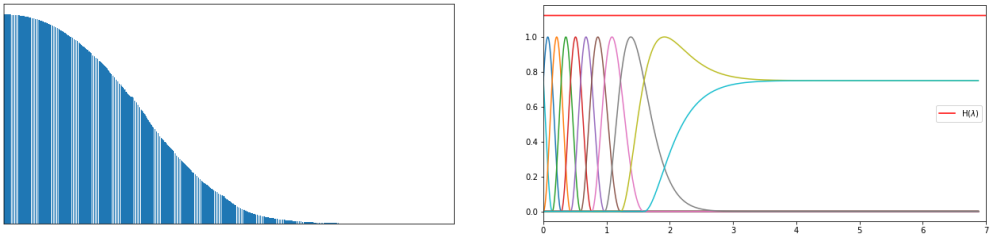
$$\hat{h}_1(\lambda) := \sqrt{\frac{3R}{8} - \sum_{m=2}^{M} |\hat{g}_m(\lambda)|^2}, \tag{11}$$

as the scaling kernel, that constructs the low-pass filter we represented in (1).

Now to find a good warping function $\omega(\lambda)$ that causes localization in the spectrum of the eigenvalues (that basically represent frequencies in the time-series signal processing domain), we can set this function to be

$$P_\lambda(z) := \frac{1}{N} \sum_{l=0}^{N-1} \mathbb{1}_{\lambda_l \leq z}, \tag{12}$$

where $P_\lambda(z)$ is the empirical spectral cumulative distribution of the graph laplacian eigenvalues. 6 shows the empirical spectral distribution of all λs of the laplacian of Minnesota graph as well as the spectrum adapted kernels. As shown in this figure, kernels get more localized and tighter where the spectral distribution is higher, and they get wider when the spectral distribution decreases.



λ λ

Figure 6: Representation of the tight graph wavelet kernels that are adapted to the spectral distribution of the laplacian eigenvalues of the Minnesota graph.