EE 378A Statistical Signal Processing Professor Andrea Montanari

Homework 2

Due on: 04/25/2022

Instructions: this homework has a numerical component. Please submit your code and the simulation results together with your solution.

Problem 1

Exercise 7.9 from S. Mallat, A Wavelet Tour of Signal Processing
Let θ be a box spline of degree m obtained by $m+1$ convolutions of $\mathbf{1}_{[0,1]}$ with itself.

(a) Prove that

$$\theta(t) = \frac{1}{m!}\sum_{k=0}^{m+1}(-1)^k\binom{m+1}{k}([t-k]_+)^m,$$

where $[x]_+ = \max(x,0)$. *Hint:* Write $\mathbf{1}_{[0,1]} = \mathbf{1}_{[0,+\infty)} - \mathbf{1}_{(1,+\infty)}$

Solution. Denote $\theta_m(t)$ a box spline of degree m. Then we want to show that

$$\theta_m(t) = \frac{1}{m!}\sum_{k=0}^{m+1}(-1)^k\binom{m+1}{k}([t-k]_+)^m \tag{1}$$

Let $v = \mathbf{1}_{[0,+\infty)}, u = \mathbf{1}_{(1,+\infty)}$. Denote their Fourier transforms $\mathcal{F}[u](f) = U(f)$ and since $v(t) = u(t-1)$ (for $t \neq 1$), $\mathcal{F}[v](f) = V(f) = e^{-i2\pi f}U(f)$. Thus,

$$\mathcal{F}[\theta_m](f) = (U(f) - V(f))^{m+1} = (1-e^{-i2\pi f})^{m+1}U(f)^{m+1} = \sum_{k=0}^{m+1}(-1)^k\binom{m+1}{k}e^{-i2\pi k}U(f)^{m+1}$$

Denoting $u^{(k)}$ the result of a convolution of u with itself k times, by the time-shifting property of Fourier transform we get

$$\theta_m(t) = \sum_{k=0}^{m+1}(-1)^k\binom{m+1}{k}u^{(m+1)}(t-k). \tag{2}$$

We can deduce $u^{(k)}$ by induction:

$$\begin{aligned} u^{(k+1)}(t) &= \int_{-\infty}^{\infty} u^{(k)}(\tau)\mathbf{1}_{t-\tau\in([0,+\infty)]}d\tau = \int_{-\infty}^{t} u^{(k)}(\tau)d\tau \\ u^{(2)}(t) &= \int_{-\infty}^{t}\mathbf{1}_{\tau\in[0,+\infty)}d\tau = [t]_+ \end{aligned}$$

Assume that $u^{(k)}(\tau) = \frac{1}{(k-1)!}[t]_+^{k-1}$ (this holds for $k=2$ as shown above) then

$$u^{(k+1)}(\tau) = \int_{-\infty}^{t}\frac{1}{(k-1)!}[\tau]_+^{k-1}d\tau = \frac{1}{(k-1)!}\int_0^{[t]_+}\tau^{k-1}d\tau = \frac{1}{k!}[t]_+^k.$$

By induction $u^{(k)}(\tau) = \frac{1}{k!}[t]_+^k$ holds for any $k \geq 2$ and plugging this into (2) gives (1).

(b) Let A_m and B_m be the Riesz bounds of $\{\theta(t-n)\}_{n\in\mathbb{Z}}$. Prove that $\lim_{m\to+\infty} B_m = +\infty$ using the fact that $\{\theta(t-n)\}_{n\in\mathbb{Z}}$ is a Riesz basis if and only if (theorem 3.4 and equation 7.9 from the book)

$$\forall\omega\in[-\pi,\pi],\quad A\le\sum_{k\in\mathbb{Z}}|\hat\theta(\omega+2\pi k)|^2\le B.$$

Compute numerically A_m and B_m for $m\in\{0,\dots,5\}$. You are welcome to use your favourite programming language.

Solution. For $k\in\mathbb{Z},\omega\in[-\pi,\pi]$

$$|\hat\theta_m(\omega+2\pi k)|^2=\left|\mathcal{F}[\mathbf{1}_{[0,1]}](\omega+2\pi k)\right|^{2(m+1)}=\left|\frac{\sin(\omega/2)}{\omega/2+\pi k}\right|^{2(m+1)}\le\begin{cases}1, & \text{if } k=0\\ \left(\frac{2}{\pi|k|}\right)^{2m+2} & \text{otherwise}\end{cases}$$

Taking the sum over k and the supremum over ω we obtain:

$$\sup_{\omega\in[-\pi,\pi]}\sum_{k\in\mathbb{Z}}|\hat\theta_m(\omega+2\pi k)|^2\quad\le\quad 1+2\sum_{k\in\mathbb{Z}_{++}}\left(\frac{2}{\pi k}\right)^{2(m+1)}\underset{m\to\infty}{\to}1.$$

On the other hand, the LHS of the above is lower bounded by $|\hat\theta_m(0)|^2=1$, thus

$$B_m=\sup_{\omega\in[-\pi,\pi]}\sum_{k\in\mathbb{Z}}|\hat\theta_m(\omega+2\pi k)|^2\to 1.$$

For the numerical computations we discretize the interval for ω and approximate the sum with the middle 2000 values. The results are presented in the table below:

m	A_m	B_m
0	1.395e-29	1.000e+00
1	2.549e-61	1.000e+00
2	6.744e-93	1.000e+00
3	2.184e-124	1.000e+00
4	8.072e-156	1.000e+00
5	3.276e-187	1.000e+00

Table 1: Problem 1(b): results of a numerical calculation of A_m, B_m

```python
import numpy as np

def theta(w, m):
    # returns \hat\theta_m(w)
    return np.power(np.abs(np.sin(w)/(w + (w == 0))), 2*m+2) + (w==0)

n_pts = 100001
K_limit = 1000
A = {}
B = {}
w_list = np.linspace(-np.pi, np.pi, n_pts)

for m in range(6):
    theta_sum_list = []
    for w in w_list:
        w_plus_2pik = w + 2.0 * np.pi*np.arange(-K_limit, K_limit+1, 1)
        theta_sum_list.append(np.sum(theta(w_plus_2pik, m)))
    A[m] = (w_list[np.argmin(theta_sum_list)], min(theta_sum_list))
```

```
    B[m] = (w_list[np.argmax(theta_sum_list)], max(theta_sum_list))

#print the results
for m  in range(6):
    print("m={}:".format(m))
    print("\tA_m = {:.3e}, achieved  at w={:.3e}".format(A[m][1], A[m][0]))
    print("\tB_m = {}, achieved  at w={:.3e}".format(B[m][1], B[m][0]))

```

Listing 1: Problem 1(b): code for numerical calculation of A_m, B_m in Python

Problem 2

Image mosaic

Exercise 7.27 from S. Mallat, A Wavelet Tour of Signal Processing

Let $f_0[n_1, n_2]$ and $f_1[n_1, n_2]$ be two square images of N pixels. We want to merge the center of $f_0[n_1, n_2]$ for $N^{1/2}/4 \leq n_1, n_2 < 3N^{1/2}/4$ in the center of f_1. For two grayscale images of your choice compute numerically the wavelet coefficients of f_0 and f_1. At each scale 2^j and orientation $1 \leq k \leq 3$ replace the $2^{-2j}/4$ wavelet coefficients corresponding to the center of f_1 by the wavelet coefficients of f_0. Reconstruct an image from this manipulated wavelet representation. Explain why the image f_0 seems to be merged in f_1,, without the strong boundary effects that are obtained when directly replacing the pixels of f_1 by the pixels of f_0. *Feel free to use whatever programming language you feel comfortable with, both MATLAB and Python have wavelet packages.*

Solution. Below is the code that solves the problem. The results are presented on figure 1. Strong boundary effects are not present since the convolution in the inverse transform smooths out the edges in the wavelet domain.

```
from PIL import Image, ImageOps
import matplotlib.pyplot as plt
%matplotlib inline

# load the images
N = 256
images = []
images.append(Image.open("./cat.png").resize((N, N)))
images.append(Image.open("./dog.png").resize((N, N)))

# get the wavelet transform  of the images
images_transform = []
for n,image in enumerate(images):
    crop = []
    for j in range(2):
        for i in range(2):
            crop += [(image.size[i]-N) // 2 + j*N]
    images[n] = ImageOps.grayscale(image.crop(crop))
    images_transform.append(pywt.wavedec2(images[n], 'db4'))


# paste the center patch of the first image's transform into the transform of the second
    image
new_coeffs = []
for scale in range(len(images_transform[0])):
    img1_transform, img2_transform = images_transform[0][scale], images_transform[1][scale]
    new_coeffs.append(img1_transform)
    if scale != 0:
        for k in range(3):
            h, w = new_coeffs[-1][k].shape
            u1 = h//4
            u2 = w//4
```

```
            l1 = h - h//4
            l2 = w - w//4
            new_coeffs[-1][k][u1:l1,u2:l2] = img2_transform[k][u1:l1,u2:l2]
    else:
        h, w = new_coeffs[-1].shape
        u1 = h//4
        u2 = w//4
        l1 = h - h//4
        l2 = w - w//4
        new_coeffs[-1][u1:l1,u2:l2] = img2_transform[u1:l1,u2:l2]

# calculate the reconstructed image
reconstr_image = pywt.waverec2(new_coeffs, 'db4')
reconstr_image -= np.min(reconstr_image)
reconstr_image /= (np.max(reconstr_image) / 255)
reconstr_image  = reconstr_image.astype(np.uint8)

reconstr_img = Image.fromarray(reconstr_image)
reconstr_img.save('./catdog.png', format = 'png')
display(reconstr_img)

```

Listing 2: Problem 2: code for center patch change

Figure 1: Center patch change in wavelet transform, original image (left), patch (middle), reconstructed image (right)