DSA5203
Visual Information processing and Interpretation

National University of Singapore
Data Science and Machine Learning Department
March 10, 2023

Assignment 1 Report

Wang Jiangyi, A0236307J
e0732857@u.nus.edu

1 Outline

The aim of this report is,

a) a better understanding of how to explain the result of Wavelet Transform;
b) explain the code implementation.

2 Interpretation of Haar Wavelet Tranform output

In lecture slide, it only gives the algorithm of Wavelet Transform. In Fourier Transform, the coefficients we achieve actually represent the 'power' of each sine (cosine) wave corresponding to different frequency for the given signal.

However, here, we can only understand thjs algorithm from,

a) Decomposition corresponds to down-sampling scheme. That is, filtering first, then down-sampling. Then, we may attain the sub-signal from different types of frequency.
b) Reconstruction corresponds to up-sampling scheme. That is, up-sampling first, then interpolating (and we can implement with convolution operator).

There are remaining 2 problems which naive up-sampling / down-sampling understanding cannot answer:

a) Why 'decomposition' and 're-construction' algorithm works?
b) How to interpret each entry of Wavelet Decomposition Vector?

In the following discussion, we will focus on these 2 questions and give answers to 'Haar Wavelet Transform'.

2.1 Theory perspective

2.1.1 Two Family of Function

In reality, we do not have access to continuous signal. What we have is the discrete sampling, which can be viewed as a **step-function**. To model this, it is trivial to introduce **Scaling Function** familiy V_0,

$$V_0 := \{f : f(x) = \sum_{k\in\mathbb{Z}} a_k \phi(x-k), a_k \in \mathbb{R}\} \tag{1}$$

Here, $\phi(x) := \mathbb{I}_{[0,1)}(x)$, which is a indicator function in interval $[0,1)$. Thus, V_0 represents all real functions which are only dis-continuous at integer points.

Similarly, we can define **Scaling Function** V_j with better resolution,

$$V_j := \{f : f(x) = \sum_{k\in\mathbb{Z}} a_k^j \phi(2^j x-k), a_k^j \in \mathbb{R}\} \tag{2}$$

Trivially, we have: $V_{j-1} \subseteq V_j$.

To achieve an efficient decomposition algorithm, ideally, we should introduce **orthogonality**.

Let's start with V_0. From definition, V_0 is strongly connected with $\phi(x)$, an indicator function in interval $[0,1)$. By simple observation, the most trivial function which is orthogonal to $\phi(x)$ is:

$$\psi(x) = \mathbb{I}_{[0,0.5)}(x) - \mathbb{I}_{[0.5,1)}(x) = \phi(2x) - \phi(2x-1) \tag{3}$$

This is the so-called, Wavelet Function. After the derivation of Wavelet Function $\psi(x)$, we can introduce the **Wavelet Function** familiy W_j, which is designed to achieve the orthognoality of V_j,

$$W_j := \{g : g(x) = \sum_{k\in\mathbb{Z}} b_k^j \psi(2^j x-k), b_k^j \in \mathbb{R}\} \tag{4}$$

By simple observation ($\langle \phi(2^j x-k_1), \psi(2^j x-k_2)\rangle = 0$ for arbitrary $k_1, k_2 \in \mathbb{Z}$), we can deduce that,

$$W_j = V_j^{\perp} \tag{5}$$

2.1.2 Decomposition

Moreover, it is easy to show that, $V_{j+1} = W_j + V_j$. Observe that, the following linear system has solution since the matrix is non-singular.

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} a_k^j \\ b_k^j \end{bmatrix} = \begin{bmatrix} a_{2k}^{j+1} \\ a_{2k+1}^{j+1} \end{bmatrix} \tag{6}$$

Therefore, based on the orthogonoality, we can further re-write the equality as follows recursively:

$$V_{j+1} = W_j \oplus V_j = ... = W_j \oplus ... \oplus W_0 \oplus V_0 \tag{7}$$

Note that, this decomposition is unique for the property of orthogonality. This decompisiton illustrates that, for arbitrary function $f_{j+1} \in V_{j+1} = W_j \oplus V_j$, we can decompose it to 2 parts:

a) average in 2 consecutive intervals, which is V_j part;
b) fluctuation in 2 consecutive intervals, which is W_j part.

Now, due to the 'exact equality', we can achieve 'decomposition' and 'reconstruction' based on some careful calculation.

2.2 Application to Haar Wavelet Transform

Now, suppose we have a sampling signal. It can be modelled by a function $f_j \in V_j$ according to our sampling frequency. Therefore, we can apply our previous discussion result and this exactly corresponds to our **Haar Wavelet Transform Algorithm**:

1. Decomposition part: given coefficients in V_j, calculate corresponding coefficients in W_{j-1}, V_{j-1}. Here, given 2 consecutive coefficients in V_j, we can construct 1 coefficient in W_{j-1} and 1 coefficient in V_{j-1} via linear combination.
Therefore, this step can be achieved by **convolution (linear combination) + down-sampling (1 calculation is redundant)**.

2. Reconstruction part: given coefficients in W_{j-1}, V_{j-1}, calculate corresponding coefficients in V_j.
Therefore, this step in equation (6) can be achieved by **up-sampling (from index k to $2k, 2k+1$) + convolution (linear combination)**.

2.3 Discussion

Up to now, we can answer the 2 questions:

a) Why 'decomposition' and 're-construction' algorithm works?
Answer: It is just one implementation of the previous decompositon, $V_{j+1} = W_j \oplus ... \oplus W_0 \oplus V_0$. Wavelet coefficients just correspond to the coefficients of $\psi(2^j x - k), \phi(x - k)$. Algorithm works based on the 'exact equality' of direct sum decomposition.

b) How to interpret each entry of Wavelet Decomposition Vector?
Answer: For coefficients of $\{V_i\}$, it can be understood as **low-resolution approximation** of sampled signal; for coefficients of $\{W_k : k = i, ..., j\}$, it can be understood as **multi-scale fluctuation** around low-resolution approximation. The different indices of coefficients correspond to different location of sample signal. Both can be viewed as the **'power' of 'wavelet'** for some scale and **translation**, which is similar to the meaning of Fourier Transform.

Moreover, we can see that, 'Haar Wavelet Transform' algorithm is just one **efficient implementation** of previous direct sum decomposition.

To check its efficiency, let us compare with the most naive case. Since we have the following decomposition,

$$V_{j+1} = W_j \oplus ... \oplus W_0 \oplus V_0 \tag{8}$$

the most naive way to achieve the coefficient is, figure out the exact solution of each coefficient of $\{V_0, W_0, ..., W_j\}$, then plug in our sampled signal coefficients $\{V_{j+1}\}$. At last, we can derive the decomposition result.

However, observation is, **there exists tremendous repeated (redundant) computation**! For example, the calculate of $\{V_0, W_0, ..., W_{j-1}\}$ will share the same computation since all of them are decomposed from

$\{V_j\}$.

To solve this issue, we can apply the idea of **'Dynamic Programming'**. That is, using **backward induction** to kick out those redundant computation at the cost of **more storage space**. To be more specific, to achieve the decomposition of equation 8, firstly we conduct: $V_{j+1} = W_j \oplus V_j$. In the LHS, W_j is what we want and we will **store V_j in memory for more efficiency**. To achieve W_{j-1}, we start from V_j and repeat this procedure until we achieve V_0.

This is what we do in this assignment, and we will **illustrate more** in the following discussion. Actually we can still **conduct some optimization** as we will discuss later.

3 Code Implementation

We discuss coding part in the following 3 aspects: coding outline, details of coding and generalization.

3.1 Outline of Coding

We construct the k-level decomposition / reconstruction wavelet transform via 1-level implementation.

1-level decomposition: given input I, we conduct convolution followed by down-sampling. Finally, we output $\{s_1, w_{11}, w_{12}, w_{13}\}$;

k-level decomposition: recursively conduct '1-level decomposition' with respect to $\{s_i\}, i = 0, 1, ..., k-1$;

1-level reconstruction: given input $\{s_1, w_{11}, w_{12}, w_{13}\}$, we conduct up-sampling followed by convolution. Finally, we do summation to achieve output I;

k-level reconstruction: recursively conduct '1-level reconstruction' with respect to $\{s_i, w_{i1}, w_{i2}, w_{i3}\}$ (to achieve s_{i-1}), $i = k, k-1, ..., 1$;

3.2 Discussion on Details

3.2.1 Boundary Treatment

If we use 'full' mode of convolve2d (**this is what we use in assignment**), then down-sampling operation should start from index 1.

If we use 'valid' mode of convolve2d, then down-sampling operation should start from index 0.

3.2.2 Optimization

Through coding process, I find that, there still exists something can be optimized in the **decomposition part** of Wavelet Transform Algorithm:

Generally, for 1-level decomposition, it can be formulated as, convolotuion followed by down-sampling. However, during this process, half of the convolution operations are wasted. Thus, we can use 'convolution with stride' to repace 'convolution followed by down-sampling'.

3.2.3 Uint8

To attain correct image (Image.fromarray()), we change data type of **ihaar** output, **from float to uint8**.

3.3 Generalization

In assignment, we assume that the image has the size $(2^n \times 2^n)$. This will greatly simplify the design of algorithm. Moreover, I find that, **small changes can be made so that our algorithm can work for arbitrary size image** $(k \times k)$. **Approach is**, padding k to the closest 2^n and conduct the previous Wavelet Transform. However, here the coefficients we achieve are no longer length k but length 2^n. Here is one simple example, in which the input size is 13×13:

```
In [74]: sig = np.random.rand(13,13)
         coef, length = gen_haar2d(sig, 2)
         rec = ihaar2d(coef, 2)[:length,:length]

In [75]: sig[0,:]

Out[75]: array([0.1006841 , 0.62233588, 0.33786554, 0.77277118, 0.98325852,
                0.50163564, 0.29727657, 0.91165509, 0.66210054, 0.78174138,
                0.93200931, 0.47161546, 0.23179266])

In [76]: rec[0,:]

Out[76]: array([0.1006841 , 0.62233588, 0.33786554, 0.77277118, 0.98325852,
                0.50163564, 0.29727657, 0.91165509, 0.66210054, 0.78174138,
                0.93200931, 0.47161546, 0.23179266])
```

Figure 1: First Row of Input and Recover Result

As shown in Figure 1, it shows the effectiveness of our generalized Wavelet Transform, which can take arbitrary size image as input.