Wavelets assignment:
fingerprint compression using wavelet packets

Academic year 2014-2015

Figure 1: Taking a fingerprint is the easy part. But what to do with it afterwards?

Introduction

The most popular application of wavelet remains, arguably, compression. One of the early uses of wavelets in the 1990's, which received wide attention at the time, was a very specific type of compression: the compression of fingerprints. It resulted in a standard called the Wavelet Quantization Standard (WSQ). Storage requirements posed a genuine issue at the time and wavelets appeared to achieve higher compression ratios than other schemes based on Fourier series. In particular, the *block* artefacts of JPEG compression are detrimental to the quality of a fingerprint, where preserving the transitions between ridges, and the endings and branches of the so-called *minutiae*, the lines that make up our fingerprint, is essential.

The observation that human fingerprints uniquely identify a person has had an enormous impact on society. Yet, this is not our main concern in this project. Instead, we ask ourselves the question: why wavelets? What is the mathematical foundation of the succesful application of wavelets in this application?

In order to answer these questions, we explore the effect of discontinuities in a signal and an image on wavelet compression and Fourier-based compression. We set out to make comparisons which are not always completely fair, but they are insightful. It turns out that a particular generalization of wavelets, called *wavelet packets*, enables better results. Underlying the succes of wavelet packets is a beautiful algorithm which singles out an optimal sparse representation among exponentially many ($> 2^N$) different choices in only $\mathcal{O}(N \log N)$ operations.

1 Getting started: point discontinuities and edges

Practical arrangements for this project are summarized in §5. Do make sure that the explanation in your report is reproducible: for example, always specify which wavelet you have used and how many decomposition levels.

1.1 What's in a discontinuity? Infinitely many frequencies

Consider the following quite discontinuous function:

$$h(x) = \begin{cases} 1 + x, & \text{if } x < 0, \\ -1 + x, & \text{if } x \geq 0. \end{cases} \tag{1}$$

Task 1.1: Sample $h(x)$ in a set of equispaced points on the interval $[-1, 1]$ and compute the discrete wavelet transform of these samples. Describe the behaviour of the wavelet coefficients. Use a number of different wavelets and count the number of 'large' wavelet coefficients, as a function of the total number of samples. What do you consider to be a 'large' coefficient?

The function is periodic on $[-1, 1]$. A reasonable set of N sampling points could be:

```
t = linspace(-1, 1, N+1);
t = t(1:end-1);
```

Note the omission of the right endpoint $+1$. This ensures that the periodic extension of the vector $f(t)$, where f is a periodic function on $[-1, 1]$, does not repeat the value $f(+1) = f(-1)$. Also, use periodic wavelets: `dwtmode('per')`.

Task 1.2: Repeat the experiment but using the FFT instead (Matlab command `fft`). How many large coefficients are there?

Question 1.3: Can you explain the difference between the results of the previous two tasks?

Question 1.4: When are some wavelet coefficients *exactly* zero?

Useful Matlab commands for the one-dimensional wavelet transform include:

```
wavedec, waverec, wfilters, dwtmode
```

As usual, type `help command` to read how the command is used.

1.2 Discontinuities in flatland: points and edges

We move up from the one-dimensional to the two-dimensional world. But we can repeat the same experiment. Let's consider a point first.

Task 1.5: Construct a matrix of zeros, but with a single entry equal to 1. Compute its two-dimensional discrete wavelet transform (see `wavedec2,waverec2`). Describe the behaviour of the wavelet coefficients and count the number of 'large' coefficients.

Task 1.6: Use the FFT instead (`fft2`).

Wavelets are provably optimal for representing point discontinuities – at least within the mathematical definition of optimality that was used in the proof. Not so for edges, however. We illustrate this effect next by studying the wavelet transform of a characteristic function of a domain.

Sampling theory in more than one dimension is heavily concerned about the Fourier transform of characteristic functions. Given a domain $\Omega \subset \mathbb{R}^2$, its characteristic function is:

$$\chi(x) = \begin{cases} 1, & \text{if } x \in \Omega, \\ 0, & \text{otherwise.} \end{cases}$$

Characteristic functions are very easy to implement. For example, the characteristic function of a circle with radius R simply checks whether $x^2 + y^2 \leq R$.

For the following two tasks, sample the characteristic function of a domain, with sample points in a bounding box such that the domain lies in its interior.

Task 1.7: Compute the wavelet transform of the matrix and count the number of large coefficients. When is a wavelet coefficient *exactly* zero?

Task 1.8: Repeat, but using the FFT (`fft2`).

After tasks 1.7 and 1.8, we have at our disposal both a wavelet and a Fourier approximation to a characteristic function. At this point, it would be instructive to interpolate these approximations in a finer discretization and see which function they correspond to on that finer mesh. This would say a lot about how both transforms treat edges in images. However, implementing this is fiddly and we continue instead in another direction: wavelet packets.

2 Wavelet packets

2.1 Description

In each step of the wavelet transform, a low-pass filter $\tilde{H}$ and a high-pass filter $\tilde{G}$ are applied to a set of N coefficients. The high-pass filter yields $\frac{N}{2}$ wavelet coefficients, which describe the high-frequency content of the signal. The output of the low-pass filter also yields $\frac{N}{2}$ coefficients[1], which describe the low-frequency content. The low-pass coefficients form the input for the next iteration of the wavelet transform. Thus, the lower half of the frequency spectrum of the original signal is further subdivided into its low and high-frequency parts (with $\frac{N}{4}$ coefficients each), while the upper half remains unchanged.

Technically, there is no reason why the filters $\tilde{G}$ and $\tilde{H}$ can not also be applied to the wavelet coefficients. Doing so means one also subdivides the upper half of the frequency spectrum of a signal in a low-frequency and high-frequency part. This leads to different basis functions in the time-frequency plane: wavelet packets. Compared to wavelets, wavelet packets can be more localized in frequency. It is a consequence of the uncertainty that they are also less localized in time.

[1] Roughly. In practice the exact number of coefficients in Matlab's wavelet toolbox may depend on the chosen treatment of the boundary. See `help dwtmode`.

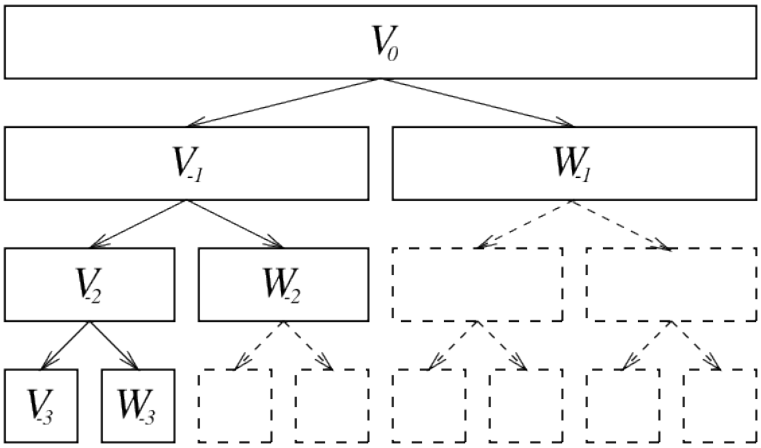

Figure 2: Function spaces spanned by wavelet packets. V_0 is the space spanned by scaling functions at the finest scale. The spaces V_{-j} correspond to scaling functions on coarser scales. The spaces W_j are spanned by wavelets on several scales. All the other spaces are spanned by *wavelet packets*. The size of each block is an indication of the dimension of the space: if V_0 is N-dimensional, then V_{-1} and W_{-1} both have dimension $\frac{N}{2}$. Blocks on lower rows have better localization in frequency, but the associated basis functions have a wider support in time.

Several possibilities now arise during the transform. For each set of coefficients that is the (downsampled) output of a filter, we can decide whether or not to apply $\tilde{H}$ and $\tilde{G}$ again. The process is always reversible, since the application of $\tilde{G}$ and $\tilde{H}$ is reversible. The case of the wavelet transform is now only a special case, where we decide to subdivide further only the low-pass coefficients and leave the high-pass coefficients unchanged. An extreme case is where we continue applying $\tilde{H}$ and $\tilde{G}$ always to all outputs. The resulting coefficients correspond to small intervals of the frequency spectrum, but large intervals in time. That is, the corresponding basis functions are highly localized in frequency, but smeared out maximally in time. In that sense, this choice is comparable to computing the discrete Fourier transform of the original signal. Many intermediate choices can be made.

The process of the wavelet packet decomposition is best illustrated by a tree structure, as shown in Fig. 2. Say the original signal lies in the function space V_0. The wavelet transform uses the basis functions of V_{-3}, W_{-3}, W_{-2} and W_{-1} to obtain a time-frequency representation. These would correspond to the nodes labelled $(3, 0)$, $(3, 1)$, $(2, 1)$ and $(1, 1)$ respectively. In general, any subtree of the full binary tree yields an alternative basis for V_0. Each leaf node of the tree is associated with a set of coefficients, and the total set of coefficients thus constructed combine into the wavelet packet transform.

Wavelet packets are also briefly described in §10.2 of the coursenotes.

2.2 The Matlab way in 1D

Familiarize yourself with the wavelet packet routines in Matlab. For example, `wpdec` and `wprec` can be used for computing a wavelet packet decomposition and reconstruc-

tion. The methods `wpsplt` and `wpjoin` can be used to 'split' or to 'join' a node in the tree. This corresponds to the choice of whether or not to apply the filters $\tilde{G}$ and $\tilde{H}$ again to the set of coefficients.

```
x = rand(128, 1);
T = wpdec(x, 4, 'db4');

% visualize the tree structure
plot(T);

% don't subdivide the node (3,7)
T2 = wpsplit(T, [3 7]);

% now y should be identical to x
y = wprec(T2);

% now T3 and T are identical trees
T3 = wpjoin(T, [3 7]);
```

You can obtain the coefficients corresponding to a leaf node (only corresponding to a leaf node) using, e.g.:

```
coef = read(T, 'data', [4 1]);
```

The coefficients can be manipulated and updated:

```
coef(10) = 0;
T = write(T, 'data', [4 1], coef);
```

And if you wonder what the leaf nodes are:

```
l1 = leaves(T);
l2 = leaves(T, 'dp');
```

The first invocation of `leaves` returns leaf nodes identified by a single index. The second way returns nodes identified by the combination of depth and position (as in the vector $[4\,1]$ used above).

Question 2.1: Try to visualize the wavelet packet basis functions corresponding to a few nodes for a chosen wavelet. For example, you can compute the inverse transform of a vector containing all zeros and a single 1 at a single location (that is, all data at all leaf nodes is 0 except a single entry somewhere). Note that for any basis $\{u_k\}$, a function represented by $f(x) = \sum_k c_k u_k(x)$ gives $f(x) = u_K(x)$ if $c_K = 1$ and $c_k = 0$ otherwise.

Note that you can get and set all coefficients using `read(T, 'data')` and `T=write(T, 'data', coef)`.

2.3 The Matlab way in 2D

The two-dimensional wavelet packet implementation is very similar. The tree is now a quadtree, i.e., each node can have up to four children. They correspond to the four subbands one obtains after applying the low-pass and high-pass filters once in the horizontal and once in the vertical direction.

For a matrix A, construct the full wavelet packet tree of depth, say, 3, using:

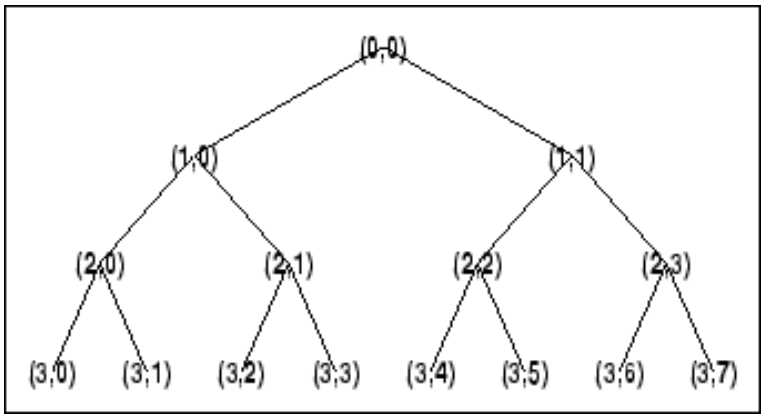

Figure 3: A binary tree of wavelet packets as visualized by Matlab. Say node $(0,0)$ contains N coefficients. Then $(1,0)$ contains $\frac{N}{2}$. Any subtree of this tree is a basis for V_0. The best-basis algorithm finds the best one.

```
T = wptree(4, 3, A, 'db4');
plot(T)
```

The first argument 4 refers to the order of the tree: it is 4 for a quadtree.

2.4 Best basis

The `wptree` command gave you a full wavelet packet tree. There are many alternative bases for the function space V_0. For example, each horizontal cross section through the full tree yields a basis. Any other cross section yields a basis too, for that matter. But which basis is the best one? To answer that question, we first need a way to compare bases. Say one basis gives rise to the coefficients $\mathbf{a} = \{a_k\}_{k=1}^N$ and another basis yields coefficients $\mathbf{b}$. We want a cost function M such that we can compare $M(\mathbf{a})$ to $M(\mathbf{b})$. A possibility is to count the number of coefficients larger than a threshold T:

$$M(\mathbf{a}) = \#\{a_k \mid |a_k| > T\}.$$

Another common cost function is the Shannon entropy:

$$M(\mathbf{a}) = -\sum_{k=1}^{N} a_k^2 \log(|a_k|).$$

An important property that the cost function should have is that of *additivity*: the result of applying M to the vector $\mathbf{a}$ should be the same as summing the costs of each entry a_k of $\mathbf{a}$. There should be a function M_e such that

$$M(\mathbf{a}) = \sum_{k=1}^{N} M_e(a_k).$$

Both examples above satisfy this property. In the case of thresholding we have

$$M_e(x) = 1 \quad \text{if} \quad |x| > T \quad \text{and} \quad 0 \quad \text{otherwise.}$$

In the case of Shannon entropy we have

$$M_e(x) = -x^2 \log |x|.$$

(For consistency, we adopt the convention that $M_e(0) = 0$.)

The best-basis algorithm now works in a bottom-up fashion as follows. We will use the Matlab-generated binary tree shown in Figure 3 to illustrate the arguments. For a quadtree, the algorithm remains the same. Note that each node in the figure corresponds to a number of coefficients. The node $(0, 0)$ represents the N coefficients of the original signal at the finest scale. Applying $\tilde{H}$ and $\tilde{G}$ and downsampling yields the coefficients of nodes $(1, 0)$ and $(1, 1)$ respectively.

First, we compute the full wavelet-packet decomposition. That is, we compute all possible wavelet-packet coefficients on all scales. Then, we iterate over the nodes of the tree, starting at the coarsest level (level 2 in the figure, since level 3 contains only leaves). The cost of having the leaves $(3, 0)$ and $(3, 1)$ can be evaluated by applying M to the corresponding coefficients and summing the two resulting values. We compare this value to the cost of the coefficients at node $(2, 0)$, i.e., of the coefficients as they were before $\tilde{H}$ and $\tilde{G}$ were applied. Depending on which cost is lower, it can be determined what the optimal way is to represent the coefficients at node $(2, 0)$: either leave them as they are, or apply the filters to get a representation with a smaller cost. This comparison can be done for all nodes on level 2.

Having determined the optimal representations on level 2, we proceed to level 1. The cost of the coefficients at node $(1, 0)$ can now be compared to the cost of the optimal representation at nodes $(2, 0)$ and $(2, 1)$. Repeating for all nodes on this level, and applying the same procedure recursively to the top, finally yields the optimal representation of the coefficients at node $(0, 0)$.

Question 2.2: Perhaps the explanation above is not sufficiently clear. It is certainly lacking some explanation. To fix this, think about the method and explain in your own words why the algorithm described above actually produces the best basis, i.e., the one that yields the smallest 'cost'. Why is the additivity property important? Why is the algorithm fast?

Task 2.3: Make your own implementation of the best-basis algorithm for binary trees and quadtrees. Note that this algorithm is also implemented by `bestlevt` and `besttree` – needless to say we won't be using those implementations. Comment your code, to make it clear that you understand what is going on. Illustrate with an example that you think illustrates the algorithm well.

Question 2.4: If you managed task 2.3, take a breath and sit back for a minute. Consider the observation that even just for binary trees, there are more than 2^N possible subtrees of the full wavelet packet tree. Each of them corresponds to a change of base and, hence, to a wavelet packet transform. You have just computed the best one in – assuming everything went well – only $\mathcal{O}(N \log N)$ operations. That's a nice trick!

3 Fingerprint compression

Why all this? Because wavelet packets are better suited to compress signals and images that have frequency content away from the zero frequency. An oscillatory function does not have a sparse wavelet transform, because any high-frequency content may result in the first $N/2$ wavelet coefficients being large and these coefficients are not processed any further in the wavelet transform. Using wavelet packets, we can zoom in on any particular frequency – much like the Fourier transform does. However, unlike

the Fourier transform, wavelet packets retain some localisation in time. Each node in the tree corresponds to a trade-off between localisation in frequency and in time. It is, by the way, not an optimal trade-off. The optimal trade off is the defining characteristic of so-called prolate spheroidal wave functions.

3.1 Compression

We proceed with the following simple transformation-based compression scheme. Compute the transform of an image, using the best basis algorithm. Find the largest coefficient, and say it has value T. Set to zero all coefficients smaller than a fraction p of T, i.e., coefficients c satisfying $|c| < pT$. If your transform was any good, you now have a very sparse vector which can be suitably coded in fewer bits than the original vector. We will not bother with the actual encoding in this project, though this is clearly a very important part of any compression codec. Reconstruct the approximation to the original signal by computing the inverse transform of the sparse vector.

Use the provided set of fingerprints (see Toledo), or take your own, to complete the following tasks.

Task 3.1: Write a routine that compresses images using the classical wavelet transform.

Task 3.2: Write a routine that compresses images using the 'best' wavelet-packet transform for that image.

Task 3.3: Quantify the difference between the two approaches by simply counting the number of remaining coefficients larger than the chosen threshold. To make the comparison a little fairer, choose threshold in both schemes that result in similar SNR (signal-to-noise-ratio) of the compression. In other words, try to match the mean squared error of the compressed image versus the original image for both schemes.

The comparison is still not entirely fair, as the best basis tree itself also requires bits to be encoded, whereas the wavelet transform uses a fixed tree.

Question 3.4: Aim for high compression ratios in both schemes, and try to describe the way in which the compressed image degrades as you discard more and more information. Can you still recognize the branch points of minutia? Can you still recognize the endpoints? What kind of compression ratio can you achieve, whilst maintaining recognizable fingerprints?

3.2 The WSQ standard

The WSQ standard specifies many properties of the wavelets that can be employed for fingerprint compression. A recent version of the standard is [2] (see document on Toledo).

Question 3.5: Consider Figure A.5 in [2]. The left panel in this figure illustrates a particular wavelet-packet basis. Which one is it? Can you recreate it? If so, compare the compression ratio you achieve with this basis compared to the best basis for a clear image of a single fingerprint.

4 Closing comments

We have omitted a few elements of a compression scheme in this project. First of all, clearly the wavelet packet tree itself has to be encoded along with the corresponding coefficients, otherwise the decoder has no way of determining the inverse wavelet

packet transform. Second, among the set of coefficients that correspond to a leaf of the tree, many coefficients are zero (or, rather, set to zero in the compression stage). One would like to encode only the non-zero coefficients, so you have to encode their locations also. For regular wavelet compression, a popular strategy is based on so-called *zerotrees*: rather than encoding where the non-zero coefficients are, it turns out to be more efficient to encode where the zeros are. Finally, encoding the actual coefficient values requires the use of a quantization technique. Quantization affects accuracy and performance of the overall scheme.

The standard for fingerprint compression is called WSQ, short for Wavelet Scalar Quantization. The particular wavelet packet tree that is used is fixed a-priori and was chosen based on many samples. Hence, it need not be encoded. It is illustrated in Figure A.5 in [2]. This tree does not differ very much from the regular wavelet transform. The standard also specifies the use of linear-phase filters.

5 Practical arrangements

We expect a report with the answers to the questions above and a brief description of the tests and experiments you have done. With each implementation, describe how you checked that the implementation is correct.

Ideally, the text of your report convinces me that you have a good understanding of the issues that are raised in all of the tasks. Include a listing of your code in an appendix, but make sure that the text itself contains sufficient information to be reproducible.

The deadline for this report is Wednesday 17 December 2014. The report may be submitted and all questions may be directed to: `daan.huybrechs@cs.kuleuven.be`. You may work alone or in pairs of two, but the latter is recommended. Contact the teacher with any questions and keep an eye on Toledo for possible updates.

Good luck!

–Daan Huybrechs

References

[1] R. R. COIFMAN AND M. L. WICKERHAUSER, Entropy-based algorithms for best basis selection, IEEE Transactions on Information Theory 38(2):p.713-718, 1992.

[2] FBI CRIMINAL JUSTICE INFORMATION SERVICES, WSQ gray-scale fingerprint image compression specification, Version 3.1, 2010.