Exercises
Computational Intelligence Lab
SS 2011

Machine Learning Laboratory
Dept. of Computer Science, ETH Zürich
Prof. Dr. Joachim M. Buhmann
Web http://ml2.inf.ethz.ch/courses/cil

Series 8, May 5th, 2011
(Image Compression with Wavelets)

Problem 1 (1D Compression and Orthonormal Basis):

Let $f(x)$ be a function expressed as a weighted sum of K basis functions $u_1(x), \ldots, u_K(x)$:

$$f(x) = \sum_{k=1}^{K} z_k u_k(x)$$

For the sake of simplicity, we will consider orthonormal basis functions (e.g. normalized Haar Wavelets) that take discrete values.Thus, we can express these basis functions as vectors:

$$\mathbf{f} = \sum_{k=1}^{K} z_k \mathbf{u}_k = \mathbf{U}\mathbf{z}$$

For a fixed basis, we want to find a good approximation for $\mathbf{f}$ using only $\tilde{K}$ coefficients, where $\tilde{K} < K$. We denote the approximation $\hat{\mathbf{f}}$:

$$\hat{\mathbf{f}} = \sum_{k=1}^{\tilde{K}} z_k \mathbf{u}_k$$

If we consider only orthonormal bases, we can formulate the compression problem as picking from the original coefficients $z_1, \ldots, z_K$ a subset $\tilde{K}$ of them which minimize the approximation error.

Let σ be a permutation of indices $\{1, \ldots, K\}$ and $\hat{\mathbf{f}}_\sigma$ the function that uses the coefficients corresponding to the first $\tilde{K}$ indices of the permutation σ :

$$\hat{\mathbf{f}}_\sigma = \sum_{k=1}^{\tilde{K}} z_{\sigma(k)} \mathbf{u}_{\sigma(k)}$$

Question: Find the permutation σ^{min} which minimizes the L^2 approximation error $\left\|\mathbf{f} - \hat{\mathbf{f}}_\sigma\right\|_2^2$:

$$\sigma^{min} = \underset{\sigma}{\operatorname{argmin}} \left\|\mathbf{f} - \hat{\mathbf{f}}_\sigma\right\|_2^2$$

Hint: Remember that the L^2 norm can be written with the help of an inner product as

$$\left\|\mathbf{f} - \hat{\mathbf{f}}_\sigma\right\|_2^2 = \langle \mathbf{f} - \hat{\mathbf{f}}_\sigma, \mathbf{f} - \hat{\mathbf{f}}_\sigma \rangle$$

Also keep in mind that orthonormal basis means that the vectors comprising the basis are mutually orthogonal (zero inner product) and have unit length:

$$\langle \mathbf{u}_k, \mathbf{u}_l \rangle = 0, \qquad k \neq l$$
$$\langle \mathbf{u}_k, \mathbf{u}_k \rangle = 1$$

Problem 2 (Wavelets for Image Compression):

In this assignment, we will apply perform image compression using wavelets. We begin with first setting up the environment on your local machine for sending your compression solution to the submission system, then we go step by step through the image compression pipeline using normalized Haar wavelets.

Submission system environment setup:

Since we are implementing a compression algorithm, the submission process to the evaluation system will remain the same as in the previous assignments. All the templates can be found here:

http://cil.inf.ethz.ch/applications/compression.

The only difference is that you have to download the `haarTrans.m` file with the function calculating the Haar basis functions.
Make sure that you do not forget to include in the final .zip file the `haarTrans.m` file along with any other helper function you have implemented.

Image Compression with Haar Wavelets:

In the following we will provide you with a brief overview of what has to be done in order to compress 2D images using normalized Haar wavelets. In our analysis we make the following assumptions:

- The images are grayscale and represented as a matrix $N_1 \times N_2$ of pixel values.
- The image matrices are square, thus $N_1 = N_2 = N$, and furthermore N is a power of two (for reasons we will explain later).

Please read the programming hints throughout the exercise to find out how to get rid of those assumptions.

The pipeline for implementing the compression function consists of the following steps:

1. Construct the basis for the one-dimensional Haar wavelet transform.
2. Apply the Haar wavelet transform to the image.
3. Perform the compression by thresholding the coefficients of the transformed image (see exercise 1).
4. Perform model selection: Which level of compression do we want?

In step-by-step fashion, we create all necessary modules for the compression part of the pipeline, which will provides the functionality for your implementation of `Compress.m`.

1. **1D normalized Haar wavelet basis**
   Since this is not a course about wavelets, we provide you with the orthonormal Haar basis. From the compression application webpage you have to download the function:

   ```
   function H=haarTrans(N)
   ```

   The `haarTrans` function provides you with the normalized one-dimensional Haar wavelet basis functions as columns in the output matrix H. Thus if you want to apply the Haar transform to a vector $\mathbf{z}$ we can just do the following multiplication in Matlab: `x = H*z ;`

   So, the first step consists of a simple call of the function `haarTrans`:

   ```
   N = size(I,1);
   H = haarTrans(N);
   ```

   Note:
   The construction scheme used in the `haarTrans` function in order to build the Haar basis only works for basis vectors which have a length of a power of two. This is why we made the second assumption at the beginning.

   Programming Hint 1:
   The width and length of an image are of course not constrained to powers of two. Thus, in order to apply the Haar basis as constructed by the given function, you have to add zeros in the right and bottom of the image (**zero-padding**) in order to make both the length and the width powers of two.

2. **2D Haar wavelet transform**
   You might be wondering how we are going to apply an one-dimensional transform to a two-dimensional signal like an image. In principle we could define the two-dimensional Haar basis functions and then directly apply them to the image. But let's stay with the one-dimensional basis that we already know and apply this to the two-dimensional image using the *standard decomposition* of an image.

   To obtain the *standard decomposition* of an image, we first apply the one-dimensional wavelet transform to each row of the image. This results in an average value along with the detail coefficients for each row. Next, treat the coefficient matrix itself as an image and apply the one-dimensional Haar transform to each column. The resulting values are all detail coefficients except for a single overall average coefficient. In algebraic terms the standard decomposition is summarized as follows:

$$\begin{aligned} \textit{Step 1 } (\text{transform rows}) &: \mathbf{Z_1} = \mathbf{IH} \\ \textit{Step 2 } (\text{transform columns}) &: \mathbf{Z} = \mathbf{H}^T\mathbf{Z_1} = \mathbf{H}^T\mathbf{IH} \end{aligned} \tag{1}$$

   Note:
   The above multiplications are valid only in the case of square images I. If you want to deal with non-square matrices you have to calculate two Haar matrices H_1, H_2 of appropriate size and then perform the algebraic calculations that are needed.

   Programming Hint 2:
   An alternative is to perform the zero-padding described above in such a way that you end up with a square padded image. Then apply the transform as described above and get rid of the extra columns and rows that were padded with zeros.

   Matrix $\mathbf{Z}$ now contains the coefficients of the Haar wavelet transform.

Figure 1: Original image (left), compressed with 10% (middle) and 1% (right) of the largest coefficients kept.

3. **Compression by thresholding**
   In Exercise 1 we defined compression as the representation of a function using fewer basis function coefficients than were originally given. We also proved for the one-dimensional case, that the reconstruction error is minimized if we pick the coefficients with the largest magnitude. It follows that the compression algorithm should follow the scheme:

   - Sort the coefficients c_{ij} in order of decreasing magnitude.
   - Define a threshold-coefficient value and set all coefficients with smaller values to zero.

   Note:
   There are many ways you can set the threshold for the compression. For example, you can define the threshold as the percentage of the largest coefficients you keep. Or, one could target a specific reconstruction error and search for a threshold that achieves this error. It is your design choice how you will implement the global thresholding of the Haar transform coefficients.

4. **Model selection**
   The choice of the threshold is directly related to the richness of the model we select for the representation. If we keep all the coefficients we achieve a reconstruction error of zero but we do not compress the image

at all. On the other hand, if we throw away all the coefficients we achieve maximal compression but at the cost of a large approximation error. Thus, there is a tradeoff between error and compression rate and you have to decide the "optimal" threshold choice.

Now, you have all the tools in order to implement compression.Fig. 1 depicts the result of the image compression when following the procedure discussed above.

Programming Hint 3:
Everything described above only holds for grayscale images, since we consider a single two-dimensional matrix in our calculations. If the image has color, then there is a third dimension, representing color components – red, green and blue (RGB) – and the matrix is three dimensional $N_1 \times N_2 \times 3$. If you want to work with each color channel independently, you can use the colon operator ":" in MATLAB to select a specific color channel:

```
I_red = I(:,:,1).
```

Thus, you have to apply the presented pipeline independently to each channel and then combine them back to form the colored image. To view your image, use the MATLAB function `imagesc(I)`.

Programming Hint 4:
Select carefully the structure `I_comp` `Compress.m` returns. Note that you will not gain anything in compression if all the zeros will be stored as doubles in the matrix. Thus, you should store only the non-zero elements of the thresholded matrix. The solution we propose is to use the matlab function `sparse()` which returns a sparse matrix. In this way you ensure that when you calculate the size in bytes of the structure `I_comp` you only count the remaining non-zero elements of the compressed image.

Image Decompression: In order to reconstruct the image you have to write a function:

```
[I_rec] = Decompress(I_comp),
```

that takes as input the compressed image from the compression step and outputs the reconstructed image using the inverse transform. `I_rec` should be an image with the same dimensions as the original image in double precision in the $[0, 1]$ interval.

Possible extensions:

1. Our approach treats the image as one block. You can try to extract non-overlapping pixel patches from the image and apply the Haar transform and thresholding on each patch independently. Then you can stitch the blocks back to form the complete image, as you did in the PCA compression assignment. You can also try to experiment with different block sizes and number of coefficients used in the compression to find the best trade-off between the approximation error and the compression ratio.
2. You may want to explore how different wavelet bases perform. You are free to experiment with other fixed bases and compare your results with the Haar wavelets that we provide in this assignment.