Lab 6 - Face detector

We propose to develop a face detector in Matlab using "Haar's characteristics" and random forests.

I. Learning base

The file available by clicking on this link proposes 1000 images of 10x10 pixels in grayscale. The first 500 were produced by cropping and then subsampling some of the images from the "Essex faces" database containing images of faces.

The next 500 were produced by downsampling blocks of pixels randomly drawn from non-face images;

Extract and install this database in your working directory.

II. Extraction of the "Haar features"

We want to produce all the Haar filters with as many positive coefficients as negative coefficients following 3 different patterns inspired by the original method of Viola and Jones seen in class:

- pattern 1: This pattern has a rectangular area of coefficients equal to 1 juxtaposed horizontally with an area of the same size having coefficients at -1.

- pattern 2: This pattern has a rectangular area with coefficients equal to 1 juxtaposed vertically with an area of the same size with -1 coefficients.

- pattern 3: This last pattern has a rectangular area with coefficients equal to 1 juxtaposed horizontally with an area of the same height but twice as wide with -1 coefficients, itself juxtaposed horizontally with an area identical to the first one.

We define all the possible filters as being the patterns of all the possible sizes and at all the possible positions inside a 10x10 pixels block.

Write a program to extract the result of the application of all possible filters on a 10x10 block. You can use the following code to help you if necessary:

```
function V=haarfeature(img)

Nofeature=0;

V=zeros(1,... nb of features ...);

% horizontal feature 1

% for all possible widths

for i=1:...

  % for all possible thicknesses

  for j=1:...

    % for all possible positions in line
```

```
        for x=1:...
           % for all positions in column
           for y=1:...
              Nofeature=Nofeature+1;
              V(Nofeature)=...;
           end
        end
    end
end

% vertical feature
...
...
...
...
...
...

% horizontal feature 2
...
...
...
...
...
...
```

How many different filters can you compute in a 10x10 block?

Apply all the filters on all the images and store the result in a table T with as many rows as images, and as many columns as filters. Add a column to this table to represent the ground truth (1 -> it is a face; 0 -> it is not).

III. Decision

Write a subroutine that, given a list of ordered values corresponding to the application of the same filter on several images, and a vector containing the corresponding ground truth, determines the threshold value on the filter that best separates the face blocks from the others. This subroutine will return the value of this threshold and the position in the list where to operate the cut.

```
function [index,se]=threshold(F,V)

...
```

Reminder: the function sortrows allows to sort an array according to a column passed in parameter.

IV. Learning

It is advisable to first develop a program to produce a decision tree, and only then the forest. You can use the data structure of your choice to represent the binary tree. One possibility (highly redundant) is to create an array as follows:

```
% Binary tree structure

nbfeatures= ... ; % see answer to question 1

height=4; % here = number of levels in the tree including the root and the leaves

nbleaves=2^(height-1);

tree=zeros(nbleaves,height+1,2);

% 1st dimension : Number of the concerned leaf

% 2nd dimension : Intermediate node

% 3rd dimension : 1 = Number of feature (or class for the leaves), 2 = threshold value

% Example tree(7,2,2) = threshold value to use on the second level node leading to the leaf 7
```

With this approach, as we only need one decision threshold at the root, all the cells of the table that represent the first level have the same value (one threshold value, one feature number).

Once the tree construction is complete, add a fourth dimension to represent a forest:

Example :

arbre(:,:,1,7) =	arbre(:,:,2,7) =

1128	3079	902	434	1
1128	3079	902	1436	0
1128	3079	502	303	1
1128	3079	502	2029	1
1128	1290	1311	37	0
1128	1290	1311	1892	0
1128	1290	1237	2606	1
1128	1290	1237	777	1

163.0000	-240.5000	-72.5000	158.0000	0
163.0000	-240.5000	-72.5000	-34.0000	0
163.0000	-240.5000	51.0000	-327.0000	0
163.0000	-240.5000	51.0000	-465.5000	0
163.0000	-162.5000	-413.5000	8.0000	0
163.0000	-162.5000	-413.5000	-21.0000	0
163.0000	-162.5000	673.5000	227.5000	0
163.0000	-162.5000	673.5000	76.0000	0

In this 7th tree, the feature drawn at random for the root is 1128. The decision threshold for this feature is 163. For the case where the observed value is lower than the threshold, then we compare the feature 1290 with the threshold 162.5, otherwise we compare the feature 3079 with the threshold -240.5 - etc. The last column of the left table represents the leaves, i.e. the decision to return (1 for a face, 0 otherwise). The last column of the right table is useless.

Give some examples of trees produced in this way following the representation given as an example.

V. Recognition and performance measures

Question 1: Test the classifier on the set of faces used for training. **Calculate the error rate obtained over several iterations and observe the variability of this performance measure.**

Question 2: Cross-validation

Split the training base into 5 parts of equal cardinality each with the same number of face and non-face images.

Perform 5 tests after performing the training on the images of 4 parts and using the images of the 5th part for performance evaluation:

	part 1	**part 2**	**part 3**	**part 4**	**part 5**
test 1	training	training	training	training	evaluation
test 2	training	training	training	evaluation	training
test 3	training	training	evaluation	training	training
test 4	training	evaluation	training	training	training
test 5	evaluation	training	training	training	training

For each test, perform several iterations of the training and **observe the variability of the error rate. Produce the average rate.**