Use of Textons to classify texture images with two kinds of classifiers

Carlos Arturo Acosta Herrera
Universidad de los Andes
Bogota, Colombia
ca.acosta977@uniandes.edu.co

1. Database

For this experiment, the database was a set of 1000 gray-scale images in jpg format, each one of them has a resolution of 640×480 pixels and focus a unique texture. The database has 40 images for each texture –therefor exist 25 different textures– and were divided in two sets: 30 images of each texture to train set, and the other 10 images were part of test set.

2. Representation

To represent the database was used a dictionary of textons created with a batch of image response to a filter bank.

A library with basic textons functions was provided by *Biomedical Computer Vision Group* of *Universidad de los Andes*. The library has 8 functions among which stand: `fbCreate` that generate a filter bank, `fbRun` that produce a cell of the same size of the filter bank where each cell contain the response of a image a one filter, `computeTextons` that create a dictionary of textons based on the return of `fbRun` and `assignTextons` that assign labels to the image pixels based on a dictionary of textons.

To create the dictionary is necessary a unique image set that contain the response to all filters of all images in train. Nevertheless, the dimension of the unique image set is bigger that the common computer memory size and a pooling is necessary. For this, the filter bank was run in each train image and later a batch of a 1% of the image was saved as a column on the fim cell (filtered images). To choose the pixel on the batch, it is assumed that the filter response distribution is uniform and therefor a random selection has a similar filter response distribution.

After that the fim cell was created, it is used to create a dictionary with the function `computeTextons`. The function used need 2 input arguments: the fim cell and a number k that define the number of textons on the dictionary. In this case, $k = 50$ because offer a good response.

3. Classification

To do the classification was used two classifier: a chi square's nearest neighbour and a random forest.

By one side, the nearest neighbour classifier need define a method that set how the distance between histograms is measured. This case the Chi square distance is used.

By other side, a random forest –in this case– only need define a number of trees in the forest. The number of tree in the classifier is 20.

4. Results

After to classify the two sets with the two classifiers, I get a confusion matrix for each case where a row show a ground true category and a column is a predicted category. This matrices can be find on the Figure 1.

The confusion matrices can let that the random forest classify works better than the nearest neighbour because is more greater the diagonal. Also can see that, logically, the performance is better on the train set that the test set, nevertheless, the performance on test set is similar to the two classifier.

The principal disadvantage of nearest neighbour classifier is that is more slower that the random forest classifier because for all new image, plus assign the textons to the image, calculate the textons histogram, too should calculate the distance between the textons histogram of the new image with all the textons histograms of train set images.

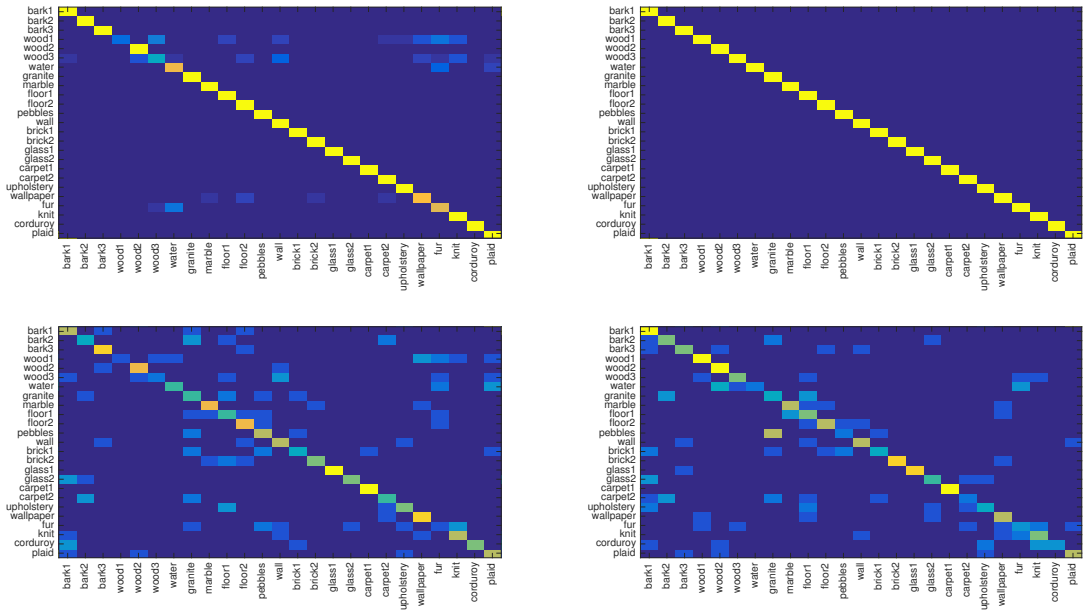

Figure 1. Confusion matrices: On top left, the confusion matrix on the train set with the nearest neighbour classifier; on top right, the confusion matrix on the train set with the random forest classifier; on bottom left, the confusion matrix on the test set with the nearest neighbour classifier; on bottom right, the confusion matrix on the test set with the random forest classifier.