Textons and classifiers

Carlos Andres Reyes Rivera
Universidad de los Andes
ca.reyes1787@uniandes.edu.co

Lina Maria Fierro Zambrano
Universidad de los Andes
lm.fierro1340@uniandes.edu.co

Abstract

The present lab is about classification methods and its used in a texture database from ponce group. The principal objective of this lab is to evaluate the classifiers k-nearest neighbour and random forest according to limitation, required time for training and results in the database. We find the better method was the Nearest neighbour with chi-square distance.

1. Database

The database for this lab comes from the page of the ponce group [1]. It has 1000 images divided in 25 texture classes. Each class has 40 different images which has 640x480 pixels, it is in gray-scale and it is in JPG format. On this lab the database was divided in two parts 750 images of train and 250 images of test. The train part has 30 images of each texture class and the test has the other 10 images.

2. Representation

In present lab, the representation of image was based in textons. So, we first created a textons dictionary with three images per class, in other words 75 train images.Additionally we used 50 textons. In the other hand, we use a filter bank, this containing Orient numbers even and odd-symmetric filters. The even-symmetric filter is a Gaussian second derivative and the odd-symmetric filter is its Hilbert transform. We try to use more images but it was not possible for computational resources. After this, assigned textons in all train images and create histograms of texton maps, this histograms was used to train classification models.

3. Classification

3.1. Nearest neighbour

Nearest neighbour is one of the most simple and fundamental classification methods. It was developed from the need to perform discriminant analysis when reliable parametric estimates of probability densities are unknown or difficult to determine[2] . The main idea of the method is classify a new image in some category based on the distance to the training data. For this purpose, it fit a Voronoi diagram during the training stage. Then the method measures the distance between the Voronoi cells and the image and assigns a category for the image. This classifier is commonly based on the Euclidean distance, however, the classifier lets to use another distances.

In this lab we represent the distance between the test images and the training with the chi-square

$$K\left(H_0, H_{0'}\right) = \sum_{i=1}^{d} \frac{\left(H_0\left(i\right) - H_{0'}\left(i\right)\right)^2}{H_0\left(i\right) + H_{0'}\left(i\right)} \tag{1}$$

and the intersection kernel distance.

$$K \cap (a, b) = \sum_{i=1}^{n} min\left(a_i, b_i\right) \tag{2}$$

In order to reach a better classification we used all of the train images to compare with the new data. For this purpose we obtained the histogram of the train images and the histogram of the new image. Then we used the chi-square and kernel intersection distance to find the image which has the closest histogram to the new data. Finally we assign the same label of the closest image to the new and repeat the same process for the next test image.

3.2. Random Forest

Random forest is a classifier method proposed by Breiman at 1999.This method used a combination of tree predictors. Each tree depends on the values of a random vector and the same distribution in each tree [3]. There are two random models: bagging and random node optimization. In the last model each node have random features. This method have some advantages like no-linear classifiers so more flexibility, the test stage is more efficient, and the over fitting problem we can resolve with randomization. But the disadvantage is the number of tuning parameters. [4]

In this lab to implement this method we used all of the train images to compare with the new data. Next, we obtain histograms in the two subsets data and predicts new labels of test images. Finally we obtained the confusion matrix and statistics like recall and precision to see the results better.

4. Results

4.1. Representation

We present some examples of textons map of train images and and their respective histograms.

Figure 1: Image 3, Class 1

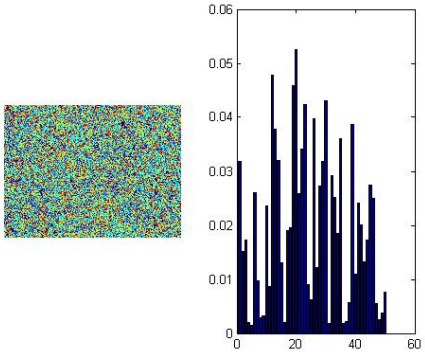

Figure 2: Map textons and histogram image 3, Class 1

The figure 1 is an original train image, this is class 1 (Bark1), figure 2 is map textons for this image and its histogram. In the other hand, figure 3 and 4 correspond to image 40 that it is class 2 (Bark2).

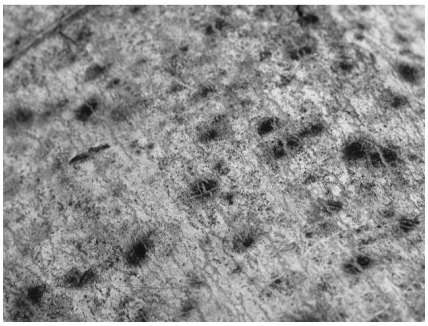

Figure 3: Image 40, Class 2

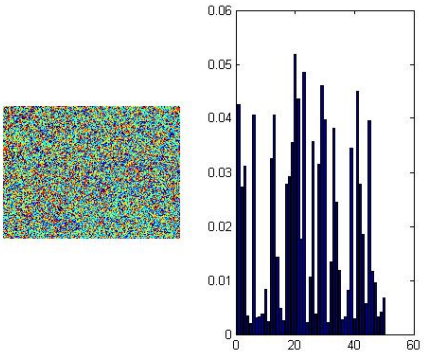

Figure 4: Map textons and histogram image 40, Class 2

It is evident that the originals images are very different but in map textons these are not evidence. Nevertheless the histograms show the differences that permit the classification.

4.2. Nearest neighbour

To implement Nearest neighbour we tried to use the Matlab function fitcknn. However this function hasn't the chi square distance or the intersection kernel,so we decided to make our own function. First of all we made the dictionary, for this we use the first 3 images of each category. This process took at least six hour and we can took more image because we have a limit compute recourse.Then, we found the textons histograms with the function assigntextons and we assigned the label of the category which they belong. This concludes the training stage and we could continued with the next stage. In the training stage the elapsed time for the chi-square distance and the intersection kernel was 787.518294 seconds.

Once we had the histogram of each one of the training images we started with the test stage. As we made

in the training, we use the assigntextons function and we obtained the histogram. To compare the histograms we first used the intersection kernel distances but this has many errors as we can see in the confusion matrix figure 7 and took 375.256802 seconds.Then we decided to probe the chi-square distances. This contrary to the other took 265.507637 seconds and obtain better results, at least find more that one category as we can see in the confusion matrix figure 6.

4.3. Random Forest

To implement Random forest in the database we used the Matlab function Treebagger and our code is based in "kawahara.ca". So for this classifier, first we train model with all train images and each of one have the corresponding label. We used 20 trees because it can provide significant information but if the number of trees increase computing time; for this reason we choose this number. An example of trained trees can be observed in figure 5.

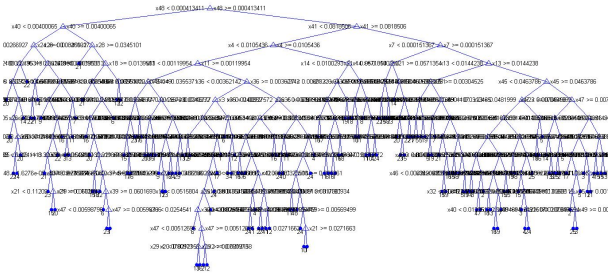

Figure 5: Tree 1 Example

Once train model made, we continued with testing stage where we used function "histc" to obtain histograms to test images and with these new available data realized the predictions. After we compared the predictions with annotations to construct the confusion matrix with Matlab function "confusionmat". This result can be found in figure 8 of appendix. Additionally, the average precision of this method was 3.7% and recall was 4%, it is evident that this statistics are very small.

With the parameters mentioned above (Number of tree, number of textons, number images in dictionary), this method have a duration time of 2137,5874 seconds including both training stage as test stage.

5. Discussion of the results

Random forest and nearest neighbor represent one of the most popular ways to work in the classification problem. This methods have been used for many researchers around the world but the results are really different. This method has a strong dependence of the descriptors which it training. In this lab, we used textons as a descriptors but we realized that they couldn't be enough to separate the classes of the ponce database. Additionally we note that this base is to specific and is focus in similar textures.

We also note that the textons are inefficient and required many computer resources. For this reason is necessary obtain only some descriptors and It would be a excellent descriptor is not enough to the ponce database.

The dictionary is the part which took more time. As all the methods need a dictionary the seconds of different is irrelevant if we take this time into account . On the other hand if we pass over the dictionary stage the more efficient method is the Nearest Neighbor with intersection kernel distance.

We can see that the results are not the best because the statistics are very small, but with this results we can conclude that the best method to classify image is Nearest Neighbor with distance Chi-Square where precision was 11% and recall 12%. The results are very small because all categories caused confusion perhaps because of the complexity of the database or obtained textons dictionary.

6. Improvements

The worst problem with this lab is the need of computer resources to make the dictionary. One possible solution for these is use another method as sketch tokens. Another trouble that we realized, is the dependency of the classifier to the descriptor. How the textons are the unique descriptor the classifier depends completely of this. But, if we use another descriptor like the shape or we try to use the techniques of detection to classify the results of the classifier could improve.

References

[1] S. Lazebnik, C. Schmid, and Jean Ponce. *A Sparse Texture Representation Using Local Affine Regions*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 8, pp. 1265-1278, August 2005.

[2] Leif E. Peterson *K-nearest neighbor* Scholarpedia, 4(2):1883.

[3] L.Breiman. *Random Forests*. Statistics Department, University of California, Berkeley, 2001

[4] P.Arbelaez. *Lecture 08: Classification*. Computer Vision, Universidad de los Andes.

Appendix

Real/Prediction	bark1	bark2	bark3	wood1	wood2	wood3	water	granite	marble	floor1	floor2	pebbles	wall	brick1	brick2	glass1	glass2	carpet1	carpet2	upholstery	wallpaper	fur	knit	corduroy	plaid
bark1	1	0	1	0	0	0	0	1	0	0	0	1	0	1	0	1	0	0	0	0	0	1	2	1	0
bark2	0	0	1	0	0	0	0	2	0	0	0	1	1	0	0	0	0	3	0	1	0	1	0	0	0
bark3	1	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	2
wood1	0	0	1	1	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	2	0	0	0	3
wood2	0	1	0	0	0	1	0	0	0	2	0	1	1	0	0	0	1	0	0	0	0	0	0	2	1
wood3	0	0	0	1	1	1	1	0	1	0	1	0	2	0	0	0	0	0	0	0	0	0	0	0	2
water	0	0	0	3	0	0	2	0	2	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0
granite	1	3	0	0	1	0	0	0	0	0	0	0	1	2	0	0	1	0	0	0	0	0	1	0	0
marble	0	0	0	1	0	0	0	0	2	2	1	0	0	0	1	0	1	0	0	0	1	1	0	0	0
floor1	0	0	1	0	0	1	1	0	0	2	0	0	0	0	2	0	0	0	0	1	0	2	0	0	0
floor2	0	0	1	2	0	3	1	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
pebbles	0	0	0	0	1	0	0	0	0	0	0	0	0	5	0	1	3	0	0	0	0	0	0	0	0
wall	0	0	0	1	0	2	0	0	1	0	2	0	3	0	0	0	0	0	0	0	0	1	0	0	0
brick1	1	0	1	0	3	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	1	0
brick2	0	0	0	0	0	0	0	0	1	0	0	0	0	0	3	0	0	0	0	1	5	0	0	0	0
glass1	1	0	0	0	2	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	3	1	0
glass2	0	1	0	0	0	0	0	0	0	0	1	2	0	1	0	2	0	0	1	2	0	0	0	0	0
carpet1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	1	0	0	0	2	0	0
carpet2	1	1	0	0	1	0	0	3	0	0	0	1	0	0	0	1	1	0	1	0	0	0	0	0	0
upholstery	0	1	3	0	0	0	0	0	0	2	0	0	1	0	0	0	0	0	0	1	0	0	0	1	1
wallpaper	0	0	0	0	0	0	1	0	0	1	1	0	2	0	3	0	0	0	0	0	1	1	0	0	0
fur	0	0	1	0	1	1	0	1	0	1	0	1	1	0	0	1	0	0	0	1	0	0	0	0	1
knit	1	0	0	0	0	1	0	3	0	0	0	1	0	0	0	0	0	0	2	0	0	0	2	0	0
corduroy	1	1	0	0	1	0	0	0	0	0	1	2	0	0	0	1	0	1	0	1	0	1	0	0	0
plaid	0	0	2	0	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	1	0	1	0	0	1

Figure 6: Confusion Matrix Nearest Neighbor with chi square distance

Real/Prediction	bark1	bark2	bark3	wood1	wood2	wood3	water	granite	marble	floor1	floor2	pebbles	wall	brick1	brick2	glass1	glass2	carpet1	carpet2	upholstery	wallpaper	fur	knit	corduroy	plaid
bark1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
bark2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
bark3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
wood1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
wood2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
wood3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
water	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
granite	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
marble	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
floor1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
floor2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
pebbles	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
wall	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
brick1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
brick2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
glass1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
glass2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
carpet1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
carpet2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
upholstery	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
wallpaper	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
fur	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
knit	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
corduroy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
plaid	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0

Figure 7: Confusion Matrix Nearest Neighbor with intersection kernel distance

Real/Prediction	bark1	bark2	bark3	wood1	wood2	wood3	water	granite	marble	floor1	floor2	pebbles	wall	brick1	brick2	glass1	glass2	carpet1	carpet2	upholstery	wallpaper	fur	knit	corduroy	plaid
bark1	8	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
bark2	0	0	0	0	0	0	0	0	2	0	2	5	0	0	0	0	0	0	0	0	0	0	0	1	0
bark3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	8	0	1	0	0	0	0
wood1	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	8	0	0	0	0	0
wood2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0
wood3	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	0	0	0	1	6	0	0	0
water	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	3	0	3	0	0
granite	0	3	1	1	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	3	0
marble	0	1	0	0	0	0	2	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	6
floor1	0	5	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	2
floor2	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
pebbles	0	0	0	3	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0
wall	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1
brick1	1	0	1	2	0	4	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
brick2	0	0	0	0	0	0	7	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	1
glass1	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
glass2	4	0	0	0	0	0	0	0	4	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
carpet1	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
carpet2	2	1	0	0	0	0	0	0	2	0	3	0	0	0	0	0	0	0	0	0	0	0	0	2	0
upholstery	0	2	0	0	0	1	0	0	1	0	1	0	5	0	0	0	0	0	0	0	0	0	0	0	0
wallpaper	0	1	0	0	0	0	0	0	0	0	1	0	0	8	0	0	0	0	0	0	0	0	0	0	0
fur	0	0	0	0	0	0	0	0	0	0	0	1	0	0	3	4	0	0	0	0	0	1	1	0	0
knit	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	8	0	0	0	0	0	0	0	0	0
corduroy	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	7	0	0	0	0	0	0	0	0
plaid	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	7	1	0	0	0	0	0

Figure 8: Confusion Matrix Random Forest