Lab 06-Textons and classifiers

Juan Carlos Leon Alcazar

March 31, 2016

1 Description of the database

The database[1] contains 1000 gray scale in JPG format, all of them have a standard 640x480 pixels resolution. Images are close ups of a given object surface, thus, containing textures found in different objects. There is a total of 17 classes [1], each contains between 30 to 90 sample images in the train set and between 10 to 30 images in the test set. Finally, there is a plain text file which specifies the naming convention for the images.

2 Methodlogy

Overall the mothodology used in this laboratory is presneten in figure ....

The original first step in the pipeline was the construction of a texton dictionary from the subset 'train'. However, due to hardware constraints, it was not possible to build this dictionary with the complete train set (750 images). Creating a texton dictionary with a set of 40 images already requires about 45 GB of RAM memory (At peak), and about 3 hours CPU time. For the experiments, a single 115GB RAM machine was available. As it was not possible to create a dictionary with the full training set, it was reduced to 85 images (5 images per class).

There is not a clear way to subsample the original training set while retaining the original data variability, in other words, it is expected that this sub sampling process should create some bias in the dictionary. However the nature of the dataset might help to mitigate this issue: textures are essentially local patterns repeated, with some variability, at the global level. Thus it can be assumed that each image contains several instances of these local patterns, that already contain some of the variability of the texture.

To further test this hypotesis...

btreve experimento con 4 histogramas T1,5Imagenes T1n30Images, T2n5Imagens,T2n30Image distancia chi-cuadrado

[1] The object classes are: Bark Wood, Water, Granite, Marble, Floors, Pebbles, Wall Brick, Glass, Carpet, Upholstery, Wallpaper, Fur, Knit, Corduroy & Plaid

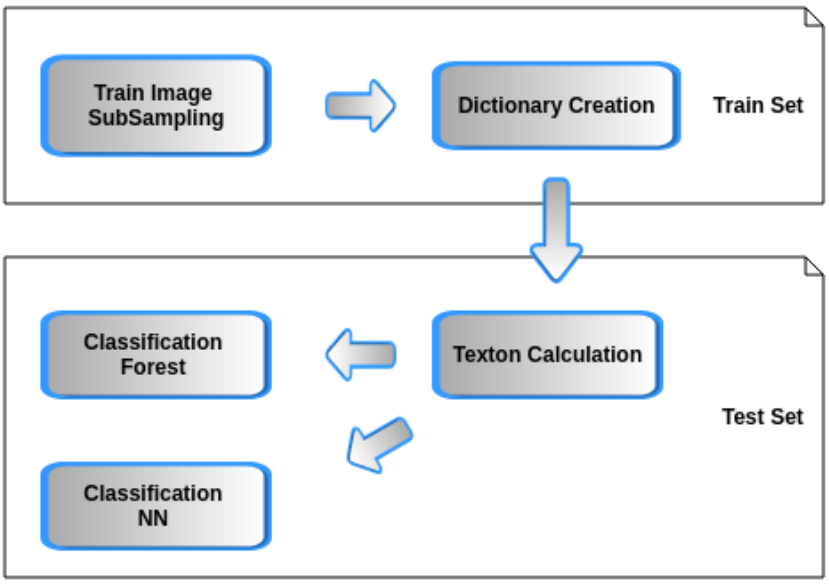

Figure 1: Elements of the proposed apporach

2.1 Textons

After selecting the initial number of training images, there remains one final parameter for the construction of the texton dictionary, namely the K for the K means. For this matter we use a number of textons given by $K = c32$ ($c = 1, 2, 3$). The explanation behind this choice is that we expected the local patterns to closely match the shape of the filter bank; This is the case of c=1 $\rightarrow$ K=32. However, not every local pattern will match perfectly one of the textons on the filterbank. This is the case of $K = 2, 3$ where the resulting clusters might contain the response information of several filters. No further values for c are explored mostly, due to time constraints. The final setup for the texton dictionary construction is the following:

- Filter Bank: default filterbank provide in the implementations 16 orientations, 2 scales
- Number of training images: 85 (5 per each class)
- Number of clusters (N): 32, 64, 96

2.2 Texton Calculation on Test Set

With the 3 Texton dictionaries built we then calculate response of every image in the test set filtered with the obtained textons. for each image we obtain 3 responses (3 dictionaries) which are then represented by means of an histogram where the number of bins depends on the value of K used for the construction of the dictionary.

Set Up	Precision	Recall
K=32	0.072	0.094
K=64	0.1609	0.4667
K=96	0.1163	0.2553

Table 1: Precision and reacall for the Nearest neigborg classfifier

Set Up	Precision	Recall
K=32,20 trees	0.1159	0.2667
K=64,20 trees	0.1552	0.3000
K=96,20 trees	0.0789	0.2000
K=32,50 trees	0.1585	0.4333
K=64,50 trees	0.1458	0.3333
K=96,50 trees	0.1111	0.2333
K=32,100 trees	0.1477	0.4333
K=64,100 trees	0.1687	0.4667
K=96,100 trees	0.1609	0.4667
K=32,500 trees	0.1638	0.6333
K=64,500 trees	0.2072	0.7667
K=96,500 trees	0.1475	0.6000

Table 2: Precision and reacall for the Random Forest classfifier

2.3 Classification

There are two methods selected for the classification of the textures

Nearest Neighbor []

Random Forest []

The first methods has a single parameter: the distance metric, as specified by the assignment, we use the Chi-square distance to choose the Nearest Neighbour.

For the second method there are more parameters, the MATLAB implementation allows to choose:

- Number of variables eligible for each decision split
- Cost of misclassifications
- Minimum number of observations per tree leaf

To allow for a maximum variability in the built trees we select the number of variables eligible for each decision split equals to the total amount of variables. Set the minimum number of observations to 1 and set no specific cost matrix for the experiments.

No adjustment is performed on the test data or the texton dictionary after being calculated.

Table 1 sumarizes the ruslts obtained for the Nearest Neighboug classifer:

Table 3 sumarizes the results obtained for the Random Forest classifier:

Set Up	Trainig time (Seconds)
K=32,20 trees	35.11
K=64,20 trees	36.30
K=96,20 trees	
K=32,50 trees	83.15
K=64,50 trees	86.37
K=96,50 trees	
K=32,100 trees	167.52
K=64,100 trees	169.77
K=96,100 trees	
K=32,500 trees	
K=64,500 trees	
K=96,500 trees	

Table 3: Trainign Time (seconds) for different configurations, times measured in a lapto computer with an Quad Core INTEL I7 and 12GB RAM

Overall both classifiers perform very poorly on the test set. The Nearest Neighbour classifier for K=32 has a performance slighty better than that of a random guess over the set of 17 classes. This behavior suggest that for K=32 the inter-class distances in the test set are very small and that the class distribution does not follow an approximate cluster pattern in the space generated by the chi-square distance metric. Thus it is very hard for any classfier to separate the clasess.

For K=64 the result are similar for both classifiers yet they barely reach a 0.2 precision. This figure suggest that both classifiers underperform mostly due to a very high number of false positives. This can be confirmed in table 4 which presents the confusion matrix for the best classifier found in this laboratory. It has a tendency to classify most sample as belonging to classes 1 and 2 (bark and wood) and mostly ignores every other class.

With the current experimental setup, this tendency to misclassify the data can be due to:

- Errors during training phase
- Selected training data does not approximate the actual variability of the test data (underfitting)

To better establish the source of error, we train and classify over the same subset (test). This experimental setup yields a precision of 1. and recall of 1.0 for the random forest with K=32. The obvious overfit scenario rules out possible errors during the training phase and further in indicates that, while the classifier can learn the patterns on the training set, this pattern are not enough to porperly classify the test set.

2.4 Trainign and execution times

Table **??** sumarizes the training times for the Random ofrest classifier

23	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	22	0	0	0	2	0	0	1	0	0	0	0	0	0	0	0
3	3	0	0	0	1	0	0	3	0	0	0	0	0	0	0	0
6	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	8	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
5	14	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
8	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	8	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
7	9	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0
15	3	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
17	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	7	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0
3	6	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
4	5	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
6	2	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
2	7	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Table 4:

The training time is mostly not an issue, most time in can be coplete in about 2 minutes or less, the only exception are the cases where the amount of trees is greater than 100. It should also be noticed thet the K used to buld the filter bank has little impact the total computation time.

Unlike the filter dictionary calculation the traingi process requeres very little RAM memory resources, usually under 400MB.

References

[1] S. Lazebnik, C. Schmid, and J. Ponce. A sparse texture representation using local affine regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1265–1278, Aug 2005.