Lab 5: Textons

Adrian S. Volcinschi M.
Universidad de los Andes
https://uniandes.edu.co/en
as.volcinschi@uniandes.edu.co

Abstract

Classification based on texture is a method that has been studied since the beginning of the century. Since it is a classification problem, this problem uses the power of supervised classifiers such as KNN, Random Forests and SVM's to train a model that can classify a new input image. The results don't look too promising to this approach of the classification problem but it doesn't mean it is the end of it or that it is garbage. This works really good for certain textures such as cars and trucks so its emphasis could be on a more specific dataset

1. Introduction

The task of classification has been around for a while now. Many approaches have been tried to fulfill this task but none have been able to accomplish it fully. One the approaches is based on texture. When we think of texture we think of something that lets us easily discriminate between two objects. One of the ways to describe texture is by using what's called textons. Textons refer to fundamental microstructures in natural images and are considered as the atoms of pre-attentive human visual perception [2]. This inspired scientists to explore textons as a way to classify objects because of their texture representation that makes them somewhat "unique". One of the first attempts at this was in 2001 by the department of computer science of Berkeley. In this attempt Leung and Malik tried to recognize visual appearances using a 3D representation of these textons [3]. This kind of thinking to classify objects has been used up to today with some modifications like local texton XOR patterns [1].

2. Materials and Methods

The dataset used for this lab was CIFAR-10 which consists of 60.000 32x32 RGB images of 10 different categories. Of this 60.000 images 50.000 were used for train and 10.000 for test. The categories are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck and each correspond to a label going from 0 to 9 respectively.

In addition to these dataset, other external tools were used such. Amongst these tools are the python functions, provided by Andres Romero, to create the textons and the plot_confusion_matrix function found in the sklearn.metrics documentation site that plots the confusion matrix in a very readble mannner.

The method to classify an image based on its texture goes as it follows:
1. Construct a filter bank for different orientations and scales.
2. Calculate the response of the first n images of the train images to each filter.
3. With these responses calculate k textons to create a dictionary.
4. Using the texton dictionary we can assign a label to each pixel of an image by looking at the closest texton it has. This is called the texton map.
5. When all train images have a respective texton map we can calculate their histograms.
6. Train a supervised classifier like KNN, Random Forest or SVM with the histograms and the corresponding labels for each image. The evaluation for this algorithm is based on the average classification accuracy (ACA).

3. Results

The first thing I decided to test was which supervised classifier was best for this task. For this I implemented three kinds: KNN, Random Forests and multiclass SVM's. The main hyperparameter for each of these models are the number of neighbors, number of trees and cost respectively. The variation of these parameters can be found on figures 1, 2, and 3.

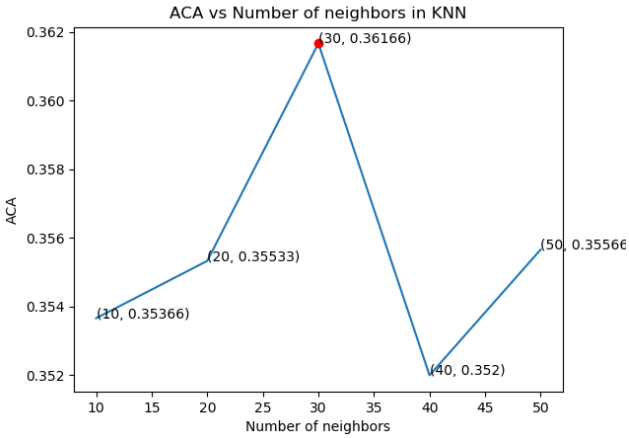

Figure 1. ACA using KNN at different k values

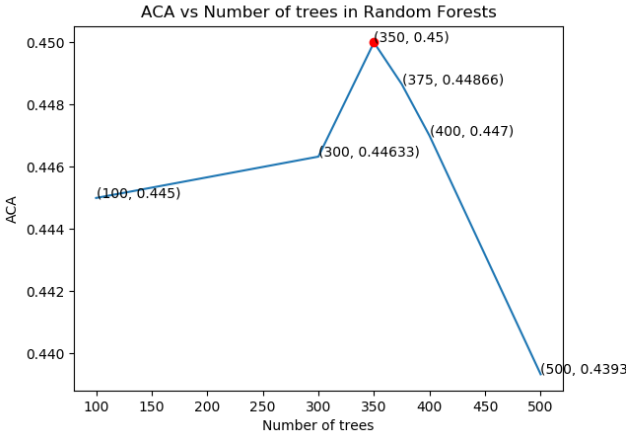

Figure 2. ACA using Random Forests at different number of trees

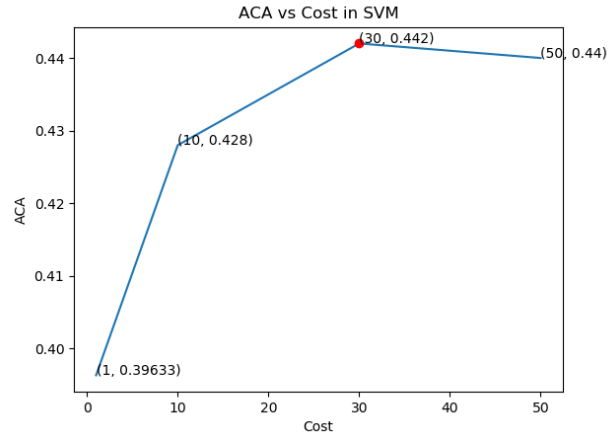

Figure 3. ACA using multiclass SVM's at different cost values

From the images shown above I decided to choose Random Forests as the best supervised classifier for this task.

Since I chose Random Forests I began to variate different parameters of them. For example, I changed the max depth of the trees as shown in figure 4.

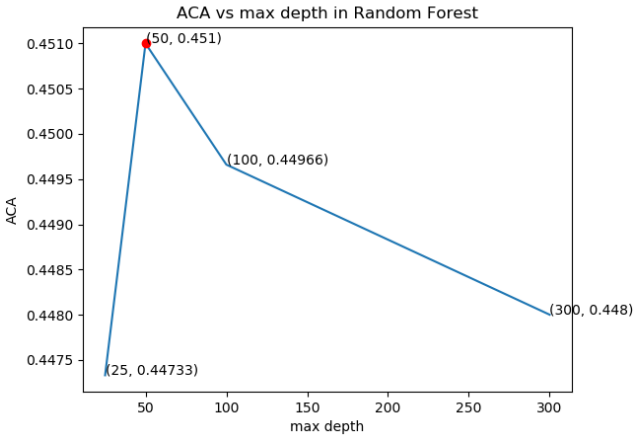

Figure 4. ACA using different values of the max depth of trees

Alongside this paramater I variated other parameters such as criterion to look at the quality of a split and minimum amount of samples to be at a leaf node whose results can be observed on figures 5 and 6 respectively.

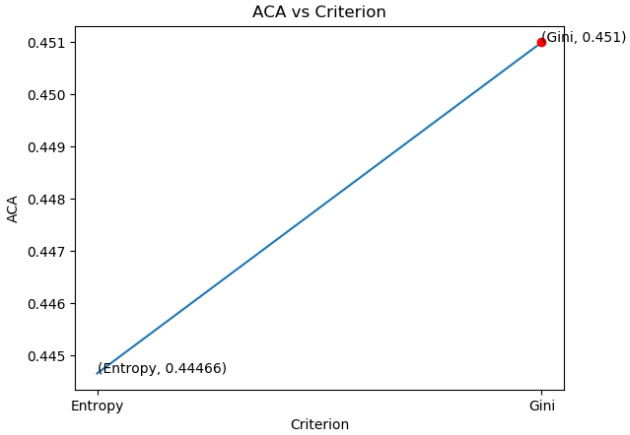

Figure 5. ACA using different quality criterion

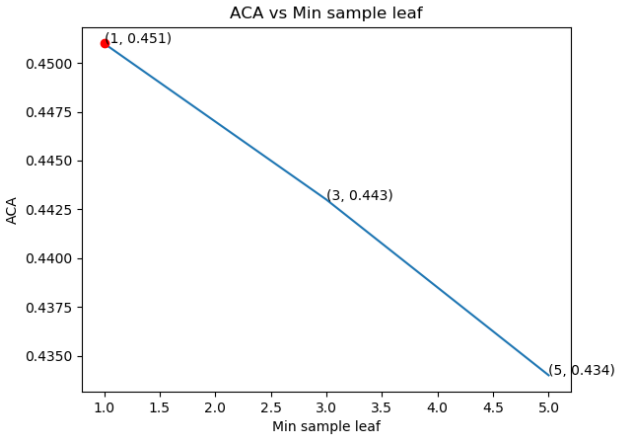

Figure 6. ACA using different values of the minimum samples needed to be a leaf

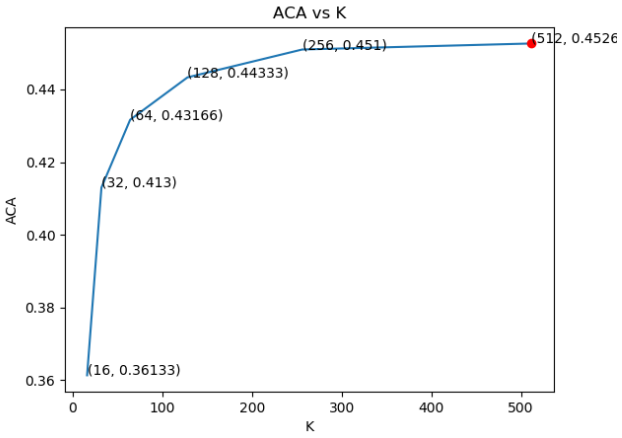

Figure 8. ACA using different values for number of textons

After all the Random Forest's parameters were set (350 trees, max depth = 50, min samples leaf = 1, criterion = "gini") I began to look at the texture parameters. First of all I began by changing the amount of images from which the textons were calculated. This process can be found in figure 7. When the number of figures used to calculate the textons was set at 200 the next thing to look forward is the number of textons to generate. I began from k = 16 up to 512. The results are shown in figure 8.

When the number of images (200) and textons (512) were chosen there were only two last paramaters to be modified, the support and startSigma of the creation of the filter bank. These two variations are shown on figures **??** and **??** respectively.

This ends up tuning our filter bank creation with a support of 2 and startSigma of 0.6.

When all the model was trained it was used to predict over the same train dataset and the ACA obtained was 1. The associated confusion matrix can be seen on figure 9. After this the same model was run against the test dataset and the ACA obtained was 0.46920 with an associated confusion matrix shown on figure 10.

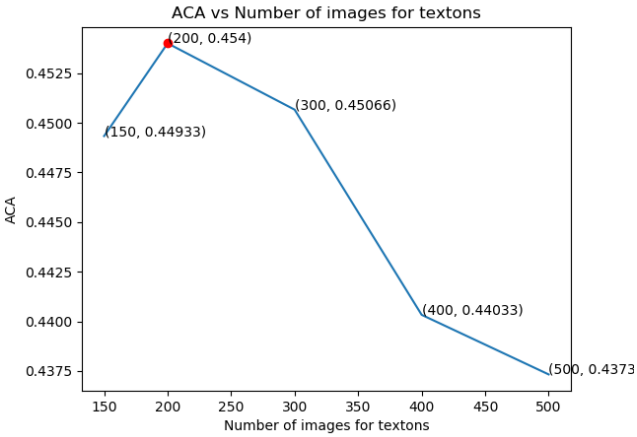

Figure 7. ACA using different amount of images to create textons

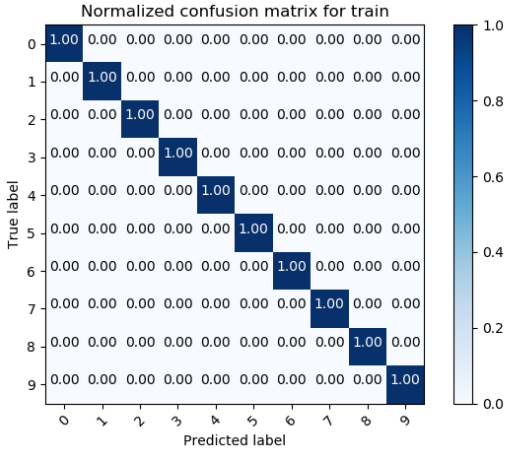

Figure 9. Confusion matrix over train dataset

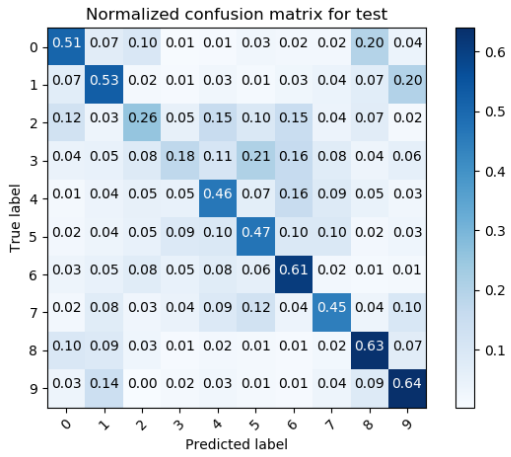

Figure 10. Confusion matrix over test dataset

Considering we were running on a 40 core server the time it took was somewhat relevant. Depending on the amount of scripts running it took more time or less but the least I was able to get was around 15 to 20 minutes. This time was for the textons of only 200 images of a whole of 50000. The time for training KNN and Random Forest was almost the same. The SVM took significantly more time than the previously mentioned. The main limitation for this approach of texture are CPU, RAM and GPU. The CPU and RAM are very obvious because there is a lot of calculations inside the K-means when the textons are greated and all the images and variables take a lot of RAM. The GPU is also a limitation since it makes the processing of images slower when the load is very high.

4. Conclusions

Looking at the results we can tell that taking the classification task with texture is not the best option. It wasn't able to obtain an ACA of 0.5 with all modifications that were done. As we can see on the figure 10 the algorithm works somewhat good with trucks, ships and frogs. On the other hand, the algorithm works really poorly for birds and cats. This can be explained by the fact that texture between animals and non-living objects like automobiles and trucks is very different. This limits the classes where non-living objects can be classified and thus reduces the error a bit. This method could be improved by augmenting the CPU and RAM of the machine where the algorithm is running so that we can calculate more textons and have a more coarse model. Additionally, trying the textons with every channel of the original RGB image could give us a bigger representation space and therefore improve the results.

References

[1] A. Bala and T. Kaur. Local texton xor patterns: A new feature descriptor for content-based image retrieval. *Engineering Science and Technology, an International Journal*, 19:101–112, 2016.

[2] B. Julesz. Textons, the elements of texture perception and their interactions. *Nature*, 290:91–97, 1981.

[3] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision*, 43:29–44, 2001.