Practicum 2: Labelling
Part 3: Improvements and Evaluation
Artificial Intelligence

Departament de Ciències de la Computació
Universitat Autònoma de Barcelona

1 Introduction

In this Part 3 of the practice we continue to work on solving the image labeling problem, combining the methods of Kmeans for color labeling and KNN for shape labeling and thus able to obtain a similar result to that in figure 1.

Figure 1: Global goal of the project.

In this Part 3 we will focus on doing experiments to measure the efficiency of the methods you have implemented. This means that you will have to analyse the optional parts that you have implemented, and you can define your own analysis measures that you will present during the oral presentation.

2 Required files

We will use the already downloaded files, but this time we will focus on the `my_labeling.py` file. Since KNN is a supervised learning algorithm, we need the images in the `Images` directory, and the learning and test sets we have in it:

1. `Images`: Folder containing the databases with the images we will use.

   Inside this folder you will find:

   (a) `Test`: Set of images that we will use as a test set.

   (b) `Train`: Set of images that we will use as a training set for classifying shapes.

   (c) `gt.json`: File with the *Ground-Truth* information of the images.

   (d) `gt_reduced`: File with complementary information on some of the images that make up the training set.

2. `test`: Folder with files for testing (In this part 3 we will no longer use them).

3. `codi`: At the root of the compressed file you will find all the necessary code and the files you need to fill (`my_labeling.py`). The files needed for the implementation are:

   (a) `utils.py`: It contains a number of features that may be useful to you, especially for the KMeans part.

   (b) `TestCases_kmeans.py`: File with which you can check whether the functions that you are programming in the Kmeans.py file give the expected result.

   (c) `TestCases_knn.py`: File with which you can check whether the functions that you program in the KNN.py file give the expected result.

   (d) `Kmeans.py`: File where you have already programmed the functions needed to implement Kmeans to classify COLOR.

   (e) `KNN.py`: File where you will need to program the functions needed to implement KNN to sort the NAME of the garment.

   (f) `utils_data.py`: It contains a number of features that may be helpful in this last part.

   (g) `my_labeling.py`: File where you will combine the two tagging methods and your enhancements to get the final names of the images.

3 Useful functions (utils_data.py)

The file utils_data.py contains a set of functions that will help you to analyze your results.

`read_dataset`: Function that loads `train` and `test` images, and *Ground-Truth* in color and shape. This function is already called at the beginning of the file my_labeling.

`read_extended_dataset`: Function that loads a subset of images for which more detailed information is available. This *Ground-Truth* is new compared to previous courses, and its objective is that you can evaluate the results in more real conditions and in a more objective way at this moment, in which you still do not know anything about computer vision, and you cannot segment the image. The information contained in this *Ground-Truth* is the following:

1. Label of the type of clothing in the picture.
2. Coordinates of a subwindow of the image that only contains a part of the piece of clothing that we are tagging. Trying to avoid the bottom of the image, the person's skin, or any other additional piece of clothing that may appear. For example, for a shirt, the subwindow will contain the central part that contains all the labellable colours.
3. Color labels of the part of the piece that is given to the subimage. These labels have been assigned by a human, not by an algorithm.
4. Background label. This field tells us if the subwindow of the image contains some background (1) or not (0). In some clothes, such as sandals, it is difficult to obtain a subwindow that captures all the colors without having to take some part of the background, most sandals have this label set to 1.

This function is already called at the beginning of the file my_labeling.

`crop_images:` Function that, given a set of images and the coordinates of the subwindows, crops the initial images so that their new size is that defined by the subwindow.

`visualize_retrieval:` Function that receives as input the ordered set of images that have been sorted as a result of a specific search and the number of images that we want to display. The function displays these images. As additional parameters you can also receive:

1. `info`: List of strings to be displayed on each of the images given by the first parameter (e.g.: `['blue jeans', 'blue flip-flops', 'blue jeans']`).
2. `ok`: List of items of type `true` or `false`. Each element refers to the corresponding image of the first parameter. The value is `true` if the predicted form and *Ground-Truth* match, otherwise it returns `false`.
3. `title`: String put by the function as global title in the figure.
4. `query`: Image for which we searched for the most similar images (only useful if this feature has been implemented).

`Plot3DCloud:` Function that receives the `Kmeans` class as input and returns the display of the point cloud that contains the image within the RGB space. Each dot appears in the color RGB corresponding to its nearest centroid.

`visualize_k_means:` Function that receives as input the class `Kmeans` and the size of the original image and returns the original image displayed using only the colors of the centroids we obtained.

4 What do we have to code?

In this third part of the practice you will code experiments to evaluate the effectiveness of your Kmeans and KNN classification methods, as well as improvements to them.

The functions to be programmed will be divided as in the following sections:

1. `Qualitative analysis functions`: Functions that allow us to visually evaluate our classifiers. You must implement at least one.

2. `Quantitative analysis functions`: Functions that allow us to numerically evaluate our classifiers. You must implement at least one.

3. `Improvements to classification methods`: Functions that will evaluate how different parameters vary can get better or worse results. You will need to implement at least as many improvements as the number of members the group is (i.e. at least one for each member of the group).

Here are some examples of features you can implement to develop analysis and improvements. Remember that you can propose and implement other functions for each of the sections; this will be evaluated positively.

4.1 Qualitative analysis functions

The aim of this analysis is to qualitatively evaluate the quality of the tags assigned with our algorithms. We'll do this by searching for specific questions, and we'll see how search results vary based on the parameters we use in our algorithms when tagging images. So it will be interesting to see how these searches would change.

`Retrieval_by_color:` Function that receives as input a list of images, the tags we obtained by applying the Kmeans algorithm to those images, and the question we ask for a specific search (that is, a string or list of strings with the colors we want to look for). It returns all images that contain the question tags we ask. This feature can be improved by adding an input parameter that contains the percentage of each color in the image, and returns the ordered images.

`Retrieval_by_shape:` Function that receives as input a list of images, the tags we obtained by applying the KNN algorithm to these images and the question we ask for a specific search (that is, a string defining the shape of clothes we want to search). It returns all images that contain the question tag we ask. This feature can be enhanced by adding an input parameter that contains the percentage of K-neighbors with the tag we are looking for and returns the ordered images.

`Retrieval_combined:` Function that receives as input a list of images, shape and color tags, a shape question, and a color question. It returns images that match the two questions, for example: Red Flip Flops. As in the previous functions, this function can be improved by entering the color and shape percentage data of the labels.

4.2 Quantitative analysis functions

`Kmean_statistics:` Function that takes as input the class `Kmeans` with a set of images and a number for `K` representing the maximum number that we want to analyse. For each value of `K=2` until `K=Kmax` it will execute the function `fit` and it will compute the `WCD`, the number of iteractions/time needed to converge etc. Finally it will visualize these data.

`Get_shape_accuracy:` Function that receives as input the tags we obtained when applying the KNN and the *Ground-Truth* of these. Returns the correct tag percentage.

`Get_color_accuracy:` Function that receives as input the tags we obtained when applying the kmeans and the *Ground-Truth* of these. Returns the correct tag percentage. Keep in mind that we can have more than one tag for each image, so you need to think about how to score if the prediction and the *Ground-Truth* partially match. In the theory class you were given some ideas for measuring the similarity between these sets.

4.3 Improvements with classification methods

`Kmeans initializations:` Within the `init _ centroids` function of the `Kmeans` class, you must implement at least two different "first" initialization methods.

`Different heuristics for BestK:` During Part 1 of this practice we programmed the heuristic `WithinClassDistance` to measure the best K, but there are other heuristics such as *inter-class distance* or the Fisher coefficient, which you can implement. You must implement at least two different heuristics.

`Find_BestK:` During Part 1 of this project we decided that to find the best K, we would use the following formula with a threshold set at 20%:

$$\%DEC_k = 100\frac{WCD_k}{WCD_{k-1}} \tag{1}$$

In this Part 3 we ask that you try to get better results by playing with the threshold value, or using other equations to determine which is the best K .

`Features for KNN:` During Part 2 of this project, to determine the nearest neighbours, we used as features the intensity of each pixel in the grayscale transformed image, and as distance, the Euclidean distance. In this Part 3 we ask you to try to get better results by trying different distances or by modifying the feature space used in Part 2. You can either reduce the size of the image, which will reduce the number of features used, or you can create your own feature spaces (e.g. mean pixel value, pixel variance, upper pixel value vs. lower pixel value, etc.).

5 Report and presentation

For the evaluation of this last part of the practice, you will have to prepare a report and make a presentation explaining the results you have put in this report.

The report must contain the following parts:

1. `Cover`: It must contain the title, subject, names and NIUs of the participants.
2. `Introduction to practice`: Brief explanation of the problem posed by the practice, the Kmeans algorithm, and the KNN.
3. `Implemented methods of analysis`: Explanation of the analysis methods you have implemented to assess how your Kmeans and KNN work. You must explain how the qualitative and quantitative methods you have programmed work. Then you need to show the results you get on the Kmeans and KNN base program. You can put all the pictures, graphs, or tables you need to explain all the well-justified experiments you've done.
4. `Improvements on Kmeans and KNN`: For each of the improvements or parameters you tried to modify:
   (a) Explanation of which parameter you are modifying.
   (b) Results we get when you change this setting or use the suggested enhancement. (Using analysis methods)
   (c) Conclusion on the results. Comparison with the results obtained without improvement and explanation of why we obtained a better / worse or similar result.
5. `Global conclusion`: Overall conclusions of the analysis, the results obtained by the implemented labeller and possible applications of what has been developed.

The presentation will follow the same structure as your report and will last 15 minutes (10 minutes of presentation and 5 minutes of questions).

6 Part 3: Submission

For the evaluation of this last part, two deliverables will be submitted. The first one, a draft of the analyses and improvements that you wish to carry out, will be delivered before 18 May at 23:55h, and the second one, which will be the final one, will be delivered before 01 June at 23:55h.

- Draft (18 May). The objective of this delivery is to be able to validate, before the presentations and the final delivery, if the evaluations of your labellers that you have designed are on the right track. A PDF file must be submitted via the Virtual Campus, detailing:

1. The qualitative and quantitative analysis functions you have decided to implement, both those already implemented and those not yet implemented.
2. A description of the experimental design you have proposed to draw results and conclusions with the analysis tools implemented.
3. The improvements or changes it has decided to implement in the K-means and KNN algorithms.
4. A description of the visualisation with which you intend to present the results.
5. The preliminary results it has obtained so far.

- Final delivery (01 July). The code you have implemented and the final report must be delivered in this last delivery of the Labeller project. Therefore, in this delivery you must upload to the Virtual Campus a zip file with the following files:

  1. Code (Kmeans.py, KNN.py i my_labeling.py).
  2. Memorandum, in PDF format.
  3. Presentation, in PDF or .ppt format.