Shape classification

Angel Lopez Manriquez

Contents

Abstract

In this document we present our approach to the problem of classifying the contour of several images, some shapes are rotated, a problem to deal with. We apply dimensionality reduction for the array (x_i, y_i) that describes the contour using hu moments, then we classify them using KNeighbors, Support vector machine and gaussian naive bayes.

We used numpy, matplotlib, tqdm scipy and IPython libraries (which are external to default python's libraries). Though the number of instances per class are few (20-30 samples) a good results were obtained.

1 Description of the problem

Object recognition is the task of automatically finding and identifying objects in an image Humans detect and recognize objects in images with extremes ease, even if objects vary in shape, size, location, color, texture, shine or are partially clogged.

The task we have to solve is the classification of the shape of several images, introduced in [1] [2] [3]. The dataset is available at: https://github.com/Ang3lino/mlnn/blob/master/projects/1Optional/hmm_gpd.zip.

The dataset includes in fact, four datasets, each one named as *bicego_data, car_data, mpeg_data and plane_data*, each instance contains a label.

The datasets has the following characteristics:

Dataset 1

- 1564 Mean of attributes.

- 7 classes.
- 210 instances.
- 30 instances per class.

Dataset 2

- 327 Mean of attributes.
- 6 classes.
- 120 instances.
- 20 instances per class.

Dataset 3

- 443 Mean of attributes.
- 4 classes.
- 120 instances.
- 30 instances per class.

Dataset 4

- 398 Mean of attributes.
- 7 classes.
- 140 instances.
- 20 instances per class.

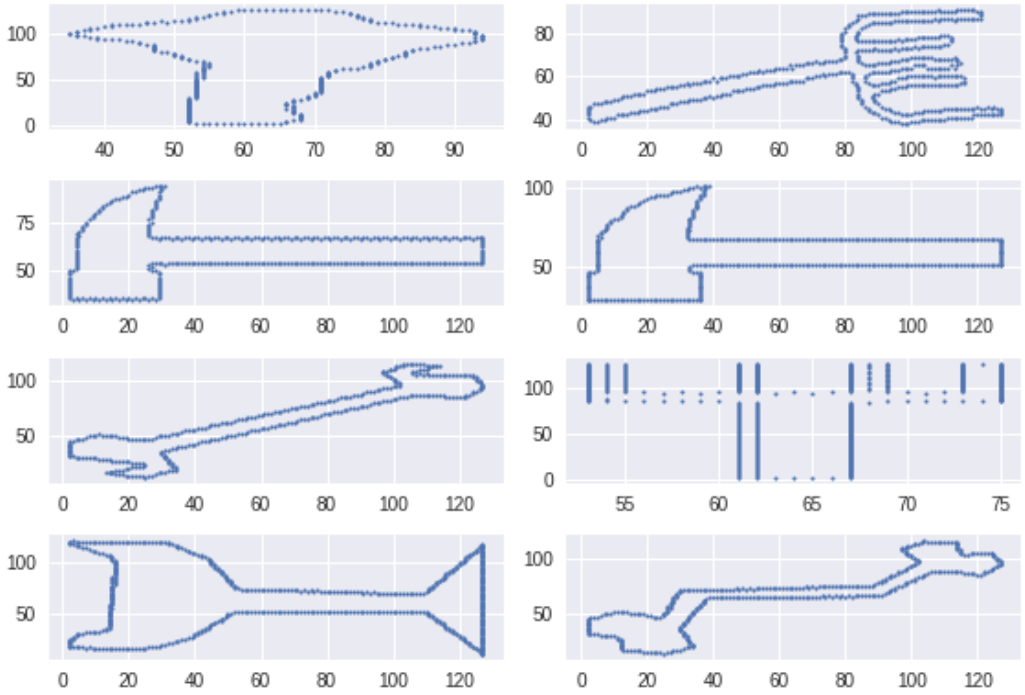
2 Description of our approach

We organized the implementation of the project according to the tasks:

1. Preprocessing of the dataset, dimensionality reduction by using hu moments.
2. Define and learn the classifiers using the training data.
3. Design the validation method to evaluate the accuracy of the proposed classification approaches, k cross validation.

2.1 Preprocessing

We use the loadmat function to store the value of the .mat file inside the variable. It loads it as a dictionary obtect where the key *x* goes to an *np.array* of *tuple* that describes a set $\{x_i, y_i\}$ representing the countour of a shape.



The problem to deal with is the dimension for each instance, which is at least 120. Shapes associated to class c were rotated. An excelent solution to solve both of the problems discussed above were the *hu moments*, given a contour it returns a feature vector of length seven. We do not loss information if the image is rotated, which is very amazing. The calculation for each coefficient is:

$$
\begin{aligned}
h_1 &= \eta_{20} + \eta_{02}) \\
h_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2) \\
h_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2) \\
h_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2) \\
h_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2) \\
&+(3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03}) \\
&(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2) \\
h_6 &= (\eta_{20} - \eta_{02})((\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2) \\
&+4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})) \\
h_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2) \\
&-(\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2)
\end{aligned}
$$

The data was split into two different sets: train and test. We use the train set to learn all the classifiers and the test set for evaluating their accuracy.

2.2 Classifiers

We use three different classifiers:

1. KNeighborsClassifier with parameters
   - **`n_neighbors = 5`**
2. GaussianNB with no parameters provided.

3. Support vector classifier with parameters
    - `gamma='auto'`

For each classifier, these were the parameters by default of the scikit-learn library. [1]

2.3 Validation

To validate our results we compute the cross-validation in the complete dataset.

As an additional validation step we computed the confusion matrices for the classifiers.

3 Implementation

All the project steps were implemented in Python. We used the loadmat function for reading the dataset, to preprocess the data we did it from scratch, almost no external library use with the exception of numpy (which improves performance over moment computation).

We illustrate how the implementation works in the Python notebook shape classification jupyter notebook

4 Results

The accuracies are shown in the notebook. The best accuracy was given by the KNeigbors classifier.

5 Conclusions

it's very impressive how mathematics help us in our job, hu moments were capable to reduce an instance of 1500 to 7, apply the trick and proceed with your machine larning algorithm!. Such representation is resistant to rotation, traslation. If there's a shape, we have much chance to classify it aproppiately.

References

[1] Krutika Bapat Satya Mallick. Shape matching using hu moments. https://www.learnopencv.com/shape-matching-using-hu-moments-c-python/, 2007.

[2] Irfan Essan Dr. Aaron Bobick. Hu moments. https://classroom.udacity.com/courses/ud810/lessons/3513198914/concepts/34988000950923.

[3] Dr. Wilfrido Gomez Flores. Reconocimiento de objetos en fotografias. https://www.tamps.cinvestav.mx/~wgomez/toptamps/tutorial.pdf?fbclid=IwAR0eiFVB5dhHXKLBw-cvy24CTpQe4DKi_-M2V34cWEZC3C8ucaoBmVACC8g, 2015.

[1] sklearn