TSIA2: Pattern Recognition

Electronic Department - option TSI, 3rd year

Rémi Giraud
remi.giraud@enseirb-matmeca.fr
2026-2027

Practical: Supervised classification
Digit recognition

1 Introduction

The objective of this practical work is to implement some shape recognition techniques in an image analysis context. More specifically, it will involve implementing discriminant analysis methods (linear and quadratic) as well as the K nearest neighbors (K-NN) method and applying them to character recognition.

1.1 Context

We want to implement an automatic character recognition method. For this, we will have a dataset of binarized training images containing the numbers 0 to 9 written in 4 different fonts, with 11 orientations and 7 scales, or 3080 images.

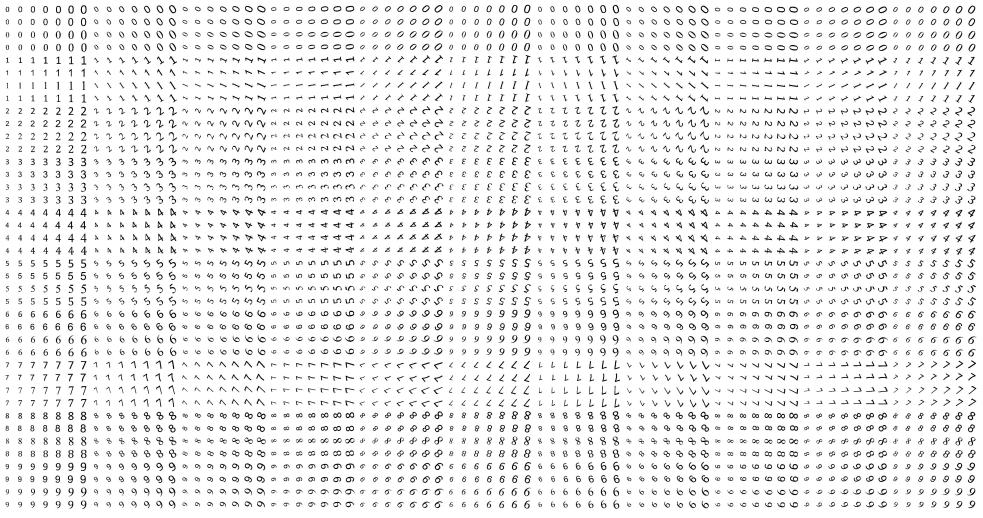

Figure 1 : Learning dataset.

These characters are the statistical individuals or forms. Each of these forms can be characterized by different descriptors. In this practical work, we will consider the first 15 Fourier descriptors.

We will use supervised classification techniques using this training data to automatically recognize new characters using Fourier descriptors. Setting up a learning system has two phases:

- learning, during which decision rules are established,
- the test, which validates the decision rules.

1.2 Generalization of digit data

The training will be carried out from "X_training.mat" which contains:

- XA: a matrix of size 3080×15 containing the Fourier descriptors for the 3080 training characters shown in Fig.1.
- YA: a vector of size 3080 indicating the class to which each individual belongs.

The test will be carried out from the "X_test.npy" file. The latter is structured in the same way as the learning base with the variable XT which contains the Fourier descriptors. The ground truth is found in the YT variable.

1.3 To do

The aim of this practical work is to complete the Python scripts provided to implement the discriminant analysis algorithms (linear and quadratic) and the K nearest neighbors (K=1 and K=N), applied to the previous data. Firstly, we will seek to recognize the shapes from a very simple description (we will take the first 5 Fourier descriptors). Secondly, we will focus on the impact of parameters (descriptors, metrics, etc.) on recognition performance.

2 Bayesian classifier

1. Write the linear discriminant analysis (LDA) and quadratic (QDA) algorithms.
2. Apply these algorithms to the test set. Evaluate the results using the confusion matrix, the correct classification rate and the Kappa coefficient.
3. Check the results obtained by comparing them with those of the *scikit-sklearn* classifier.

   In this library, all classifiers generally follow the same principle, with a call to the `fit` method to train the model, and a call to the `predict` method to generate predictions from that model:

   ```
   clf = Classifier([hyperparameters])
   clf.fit(XA, YA) #to train the model
   YT_pred = clf.predict(XT) #to get the predictions
   ```

   Classifications by discriminant analyzes are available via classes:

   ```
   sklearn.discriminant_analysis.LinearDiscriminantAnalysis
   sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis
   ```

3 K-nearest neighbors method

1. Write the classification algorithm using the K nearest neighbors method, first for K=1 then for any K=N.
2. Evaluate the classification results obtained based on K.
3. Compare the performances obtained with Euclidean and Mahalanobis metrics.
4. Compare the results to those of the *scikit-sklearn* classifier.

   The classification by the method of the K nearest neighbors is available with the class:

   ```
   sklearn.neighbors.KNeighborsClassifier
   ```

5. Modify the script to see the influence of the number of Fourier descriptors on classification performance. Conclude.
6. Then change the number of training data, taking care to reduce the amount of data equally for each class. Observe the evolution of performance and conclude.
7. What would you suggest to increase classification performance?

4 Application to speed recognition on road signs

We can try to apply our trained digit classifiers on the road sign images we segmented in the first practical. Here are the steps of the whole process:

1. Apply full HCT and selection of the inner circle to extract the digits area.
2. Extract the circle to create an image only containing the digits.
3. Find a way to separate each digit.
4. Binarize the images and compute their Fourier descriptor.
5. Try the classifiers on each digit to recognize the speed limit.

5 Bonus: Support vector machines

There are several implementations of the SVM (Support Vector Machine) classifier. We will use the following class, which makes it easy to change the type of kernel used: `sklearn.svm.SVC`

1. Apply this classifier to previous data and evaluate performance on test data.
2. What hypothesis do you make to explain this performance?

We now consider another set of data, corresponding to face images (400 images of 64x64 size from 40 different people). The dataset can be loaded with:

```
faces = sklearn.dataset.fetch_olivetti_faces()
```

3. Separate the dataset into training and testing data. For this we can use the class:

   `sklearn.model_selection.StratifiedShuffleSplit`

4. Train the classifier on training data and evaluate performance on test data.

6 Bonus: Bags of visual words

In this part, we propose to implement a classification method by bags of visual words (or *bags of features*) to from *HOG* features densely calculated on images. The dataset used is composed of 2686 color images divided into 8 classes according to the type of scenes they represent. The *scenes.py* file contains some functions to load the dataset and calculate the *HOG* features densely on an image. The objective is to complete the file so that it performs the following steps:

1. Calculation of HOG characteristics for all images.
2. Separation into two sets: 75% learning and 25% testing.

3. Construction of visual word vocabulary by k-means from the training set.

4. For each image, calculation of its histogram of visual words.

5. Classification of images from these histograms. In particular, you can use the different classifiers seen during the session and compare the results obtained.