ROB311 - TD4 - SVM Digit Recognition

BRAMBILLA Davide Luigi - GOMES DA SILVA Rafael

Octobre 7, 2019

In order to execute the program the command to run is: *python3 TD4.py*

1 Introduction

Supervised learning is a form of learning in which for a given set of inputs, the value expected for the output is already known. In this way, it is possible to map a learning function that will create the image of the inputs x_i to the outputs y_i.

Problems involving Supervised Learning can be solved by regression (when the dataset has continuous outputs), or by classification (when the dataset has discrete and categorical outputs). This report will present the results obtained for the SVM classification technique as well as a data simplification method used to improve the calculation time of the proposed algorithm.

1.1 Support Vector Machines

Support Vector Machine (SVM) is a supervised learning method used in machine learning for classification problems. This method takes as inputs a set of data and classifies it into two or more different classes, mapping the entries into the multidimensional space and using regression to find a hyperplane that best separates the input classes, as shown in the example of the Figure 1. Once the support vector machine has been trained, it is able to evaluate new entries using the hyperplane. However, because it is a supervised learning method, it is necessary to use a labled training input to build the mathematical model of hyperplanes, so the SVM can be used to classify new data effectively.

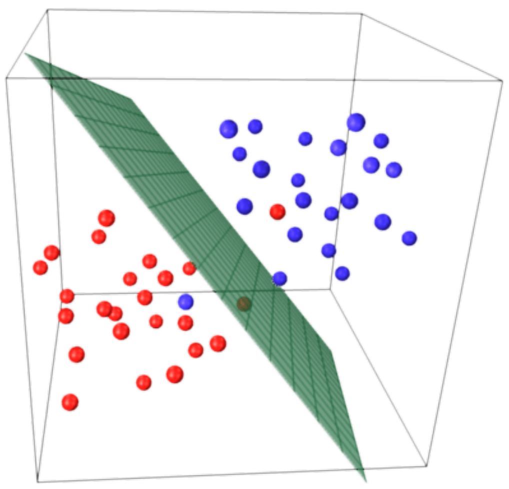


Figure 1: SVM example in the 3D space

1.2 Histogram of oriented gradients

In image processing, we call "descriptors" a vector of features that facilitate the processing of an image, considering information that could be comparable by some similarity or dissimilarity metric. The Histogram of oriented gradients (HOG), as its name implies, is a descriptor represented by a histogram vector that contains the intensity of gradients in certain directions for the sub-blocks of an image.

The algorithm is based on the idea that the shape and appearance of an object can be described by the intensity of the gradients. The Figure 2 shows an example of an image being represented using the HOG descriptor.

To generate the HOG descriptor, the image is divided into blocks and cells, each block containing a number of cells, and each cell containing a number of pixels of the image. After subdividing the image, each block counts the number of gradients that point to a defined amount of directions.

With this procedure, it is possible to turn the original image into a representation that captures the basic structure of the original image, but in a simplified way.

1.3 MNIST dataset

The Modified National Institute of Standards and Technology database (MNIST database) is a data set composed of hand-digit gray-scale images of 28x28 pixels. It contains 60,000 training images and 10,000 test images, and is widely used in supervised learning. The Figure 3 shows an example with elements present in this database.

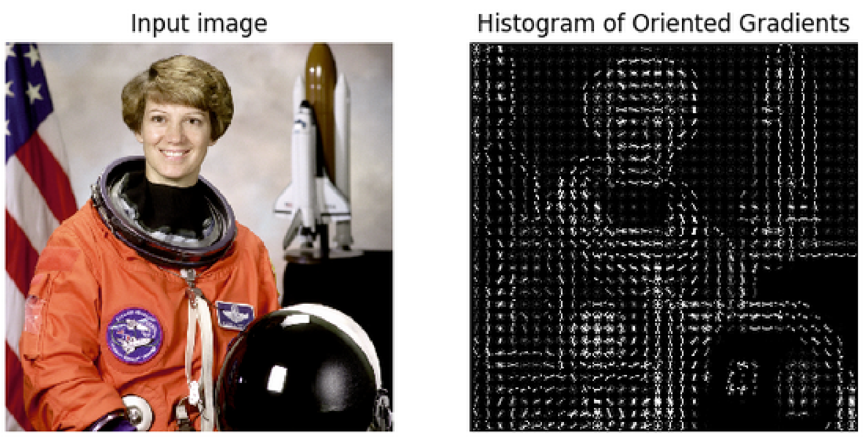


Figure 2: MNIST dataset example with the differnet handwritten samples.
Source: `https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_hog.html`

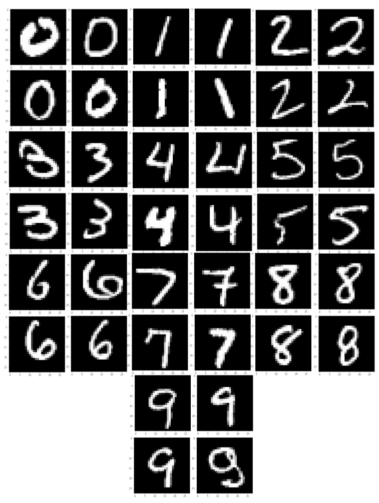


Figure 3: MNIST dataset example with the differnet handwritten samples

2 Objective

The objective of this TD was to implement a Support Vector Machines with Python in order to recognize handwritten numbers from the MNIST dataset.

3 Procedures

The data set

The structure of the MNIST data set used in this project is organized as described below:

- Each line has 785 columns, each one corresponding to one image
- The first column corresponds to the label of the image sample, that means, numbers between 0 and 9;
- The other 784 columns corresponds to each pixel of one image sample;
- The image pixels (i, j) of each image are avaiable in a 1D vector:
  $[(1,1), (1,2), (1,3) \ldots (28,26), (28,27), (28,28)]$

In order to implement the SVM digit algorithm, three helping functions were created in order to treat the data set (*separete_data()* and *convert_to_hog()*) and present the results of the algorithm (*print_results()*).

3.1 Function *separate_data(file_name)*

This function (Code 1) takes as a paramter the .csv file containg the MNIST dataset and returns the labels and the values of the pixels of the image.

Code 1: Function used to extract the labels and image features from the MNIST dataset

```
def separate_data(file_name):
   data = pd.read_csv(file_name)
   labels = np.array(data.iloc[:,0])
   x_data= np.array(data.iloc[:,1:(shape(data)[1])], "int16")
   DataFeatures = x_data
   return labels, DataFeatures
```

For this project, the function *separate_data()* function was used to extract the *y_test* and *y_train* values, and also the *x_mnist_test* and *x_mnist_train* values from both the *mnist_test.cvs* and *mnist_train.csv* datasets.

```
y_test, x_mnist_test = separate_data("mnist_test.csv")
y_train, x_mnist_train = separate_data("mnist_train.csv")
```

However, since the *x_mnist_test* and *x_mnist_train* vectors were too big, the time needed to execute the SVM algorithm was too high. In order to solve this problem, we used the HOG transformation to simplify the image characteristics and thus treat a data set with a size inferior compared to the original dataset, as it will be shown on the item 3.2.

3.2 Function *convert_to_hog(x_data)*

hog(image, orientations, pixels_per_cell, cells_per_block, visualize)

- **image**: Input image;
- **orientations**: Number of orientation bins;
- **pixels_per_cell**: Size of a cell;
- **cells_per_block**: Number of cells per block;
- **visualize**: If true, returns also an image of the HOG vectors.

```
def convert_to_hog(x_data):
   hog_list = []
   for i in x_data:
      hf = hog(i.reshape((28, 28)), orientations=9, pixels_per_cell=(4, 4), cells_per_block=(7, 7),
          visualize=False)
      hog_list.append(hf)
   x_to_hog = np.array(hog_list, 'float')
   normalize(x_to_hog)
   return x_to_hog
```

Considering the paramters used, as shown in the Code 3.2, we'll be able to simplify our image in 7 blocks with 9 histogram oriented gradient information, yielding, for each image, a descriptor vector of size 7x7x9 = 441.

For instance, the original dataset was represented by a 2D array of shape (60000, 784), while the dataset after the HOG transformation has a shape of (60000, 441).

3.3 Function *print_results(y_test,pred)*

The function *print_results()* calls the methods of sklearn to present the accuracy and the confusion matrix for the prediction done using the the SVM clasifier in the MNIST dataset.

Code 2: Détection de bords par convolution avec I_x en utilisant la methode filter2D

```
def print_results(y_test,pred):
   print ("################ ACCURACY RESULTS ################")
   print (metrics.classification_report(y_test, pred))

   print ("\n################ CONFUSION MATRIX ################")
   print (confusion_matrix(y_test,pred))
```

4 Results

In order to analyse the results of the classifier implemented, the accuracy of the digit detection for each label of the dataset was done using the *metrics.classification_report()* of the *sklearn* library, and the confusion matrix for the labels of the dataset was done using the *confusion_matrix()*, also from the *sklearn* library.

SVM classifier with the original data set

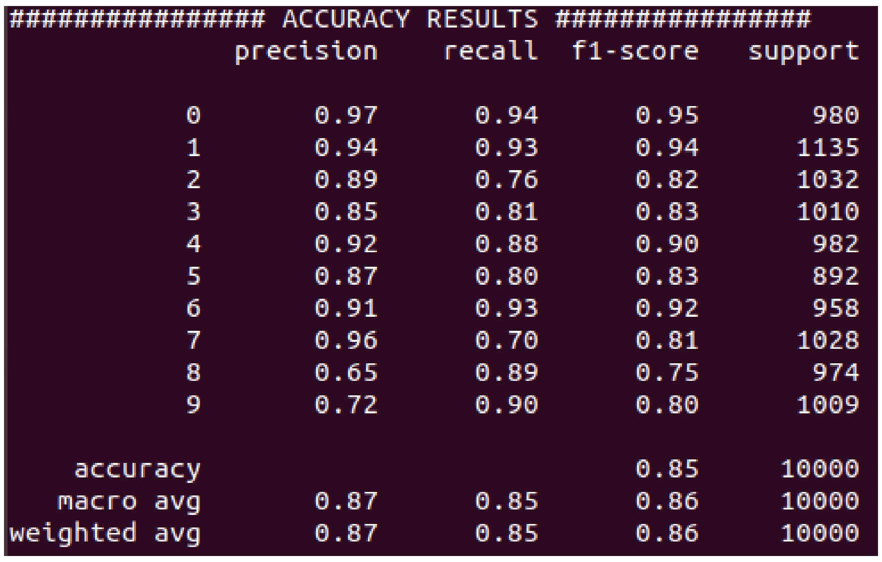
For the application of the SVM classifier using the original data set there was a convergence warning related to the size of the dataset:

```

ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
```

A possible solution found to fix this warning was to add the max_iter argument to the function *svm.LinearSVC()*. However, the warning persisted even for big values for this argument. For instance, for a max_iter=10000 the algorithm took 15min49s to complete, and for max_iter=80000 the algorithm took 1h33min38s to complete.

The mean accuracy for the SVM classifier with the original data set was 0.85. The Figure 6 shows with more details the accuracy values for each label identifier, while the Figure 7 shows the confusion matrix for this analysis case.

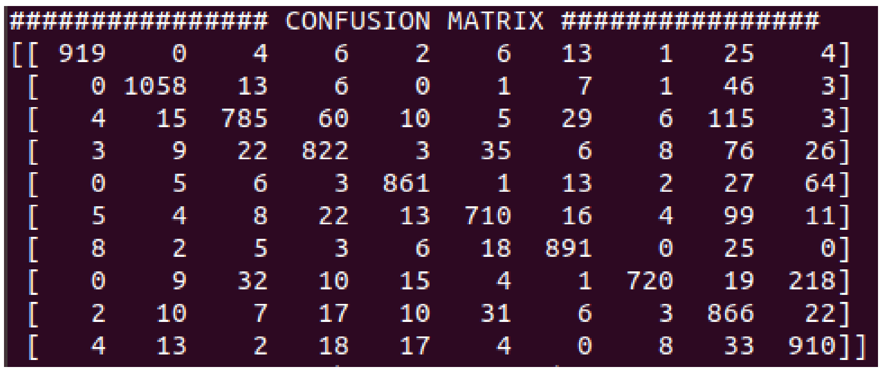


```
################ ACCURACY RESULTS ################
              precision    recall  f1-score   support

           0       0.97      0.94      0.95       980
           1       0.94      0.93      0.94      1135
           2       0.89      0.76      0.82      1032
           3       0.85      0.81      0.83      1010
           4       0.92      0.88      0.90       982
           5       0.87      0.80      0.83       892
           6       0.91      0.93      0.92       958
           7       0.96      0.70      0.81      1028
           8       0.65      0.89      0.75       974
           9       0.72      0.90      0.80      1009

    accuracy                           0.85     10000
   macro avg       0.87      0.85      0.86     10000
weighted avg       0.87      0.85      0.86     10000
```

Figure 4: Accuracy for the prediction done with the dataset

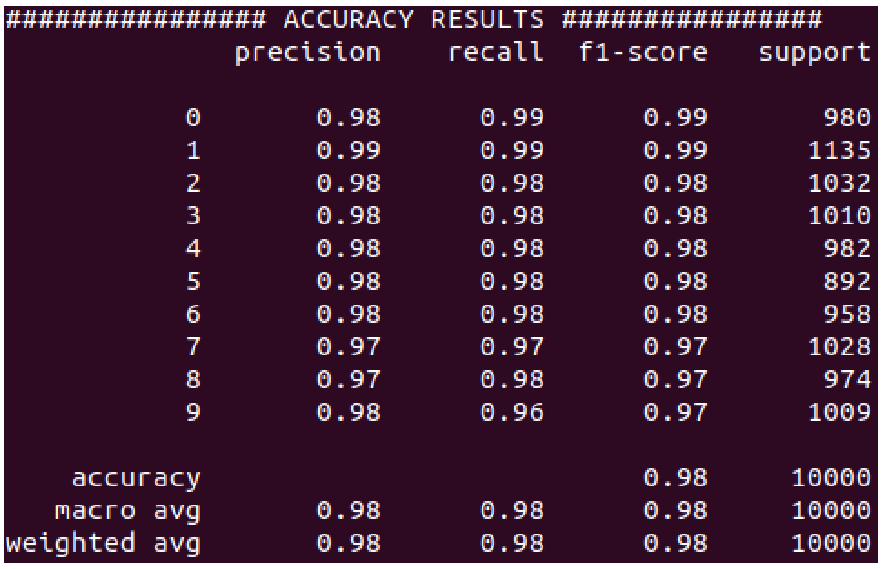


```
################ CONFUSION MATRIX ################
[[ 919    0    4    6    2    6   13    1   25    4]
 [    0 1058   13    6    0    1    7    1   46    3]
 [    4   15  785   60   10    5   29    6  115    3]
 [    3    9   22  822    3   35    6    8   76   26]
 [    0    5    6    3  861    1   13    2   27   64]
 [    5    4    8   22   13  710   16    4   99   11]
 [    8    2    5    3    6   18  891    0   25    0]
 [    0    9   32   10   15    4    1  720   19  218]
 [    2   10    7   17   10   31    6    3  866   22]
 [    4   13    2   18   17    4    0    8   33  910]]
```

Figure 5: Confusion matrix for the prediction and the labels of the test data set

SVM classifier with the data set using the HOG descriptor

For the application of the SVM classifier using the HOG descriptors of the image, it is possible to obtain a processing time of 2.77*s*, with a mean accuracy of 0.98. The Figure 6 shows with more details the accuracy values for each label identifier, while the Figure 7 shows the confusion matrix for this analysis case.

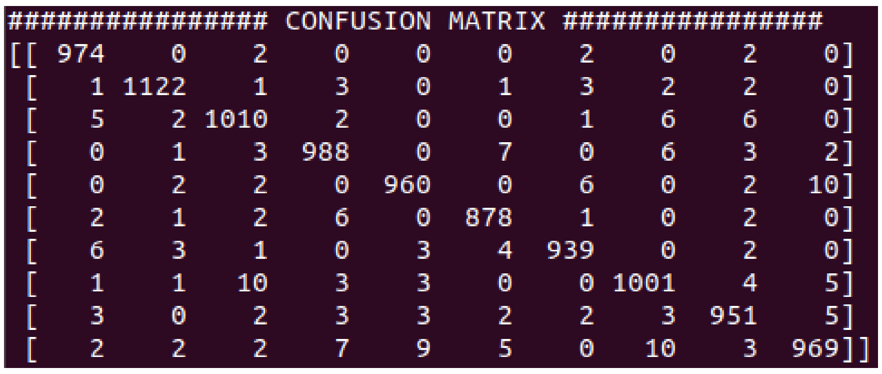


```
################ ACCURACY RESULTS ################
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       980
           1       0.99      0.99      0.99      1135
           2       0.98      0.98      0.98      1032
           3       0.98      0.98      0.98      1010
           4       0.98      0.98      0.98       982
           5       0.98      0.98      0.98       892
           6       0.98      0.98      0.98       958
           7       0.97      0.97      0.97      1028
           8       0.97      0.98      0.97       974
           9       0.98      0.96      0.97      1009

    accuracy                           0.98     10000
   macro avg       0.98      0.98      0.98     10000
weighted avg       0.98      0.98      0.98     10000
```

Figure 6: Accuracy for the prediction done with the dataset



```
################ CONFUSION MATRIX ################
[[ 974    0    2    0    0    0    2    0    2    0]
 [    1 1122    1    3    0    1    3    2    2    0]
 [    5    2 1010    2    0    0    1    6    6    0]
 [    0    1    3  988    0    7    0    6    3    2]
 [    0    2    2    0  960    0    6    0    2   10]
 [    2    1    2    6    0  878    1    0    2    0]
 [    6    3    1    0    3    4  939    0    2    0]
 [    1    1   10    3    3    0    0 1001    4    5]
 [    3    0    2    3    3    2    2    3  951    5]
 [    2    2    2    7    9    5    0   10    3  969]]
```

Figure 7: Confusion matrix for the prediction and the labels of the test data set

Conclusion

Analysing the result for both scenarios for the application of the SVM classifier implemented, we can conclude that the results were good since the overall accuracy for each method had a value over 0.8.

However, the method using the HOG descriptor of the image showed a better performance compared to the method using the original dataset. For instance, comparing the time of execution for each case, we can see that the method with the HOG data set analysis has a high speed of computation and presents an accuracy superior than the method using the original dataset. The reasons for this difference in the results might be due the warning message receveid, since in order to generate better models of hyperplans the algorithm would have need more iterations to evaluate more data.