Optimization and Data Analytics Project: A Comparison Between Classification Schemes

Davide Galli, *Aarhus University* — November 2020

Abstract — Image Classification consists of categorizing and labelling groups of pixels into different classes. To perform this task, many different algorithms have been proposed. In this paper we define and compare a subclass of them which makes use of clusters. Results show that the classification accuracy increases with the number of centroids, but with a downside in terms of execution time. PCA can improve the execution time without affecting the success rate. The whole study has been performed in a MATLAB environment due to its native approach to matrixes.

Index Terms **— Nearest Class Centroid, Nearest Subclass Classifier, Nearest Neighbor, PCA, MNIST, ORL**

NOMENCLATURE

NCC	Nearest Class Centroid
NSC	Nearest Subclass Classifier
NN	Nearest Neighbor
PCA	Principal Component Analysis
MNIST	Mixed National Institute of Standard and Technology; used as MNIST database of digits
ORL	Olivetti Research Laboratory; used as ORL database of faces

I. INTRODUCTION

IMAGE Classification is the process of categorizing and labeling groups of pixels, it is a complex procedure that may be affected by many factors. To classify a set of data into different classes, the relationship between the data and the classes into which they are classified must be well understood. Cluster based algorithms have been proved effective in this task, since they exploit the assumption that samples naturally form groups based on similarity criterion. Each sample is indeed classified according to its closest cluster. However, there are some critical choices where at current time no general rule is provided, e.g. selecting the number of clusters. Due to that, different algorithms make different choices, thus leading to different performances. Increasing the number of clusters into which train data are classified, up to the limit condition of one cluster per data point, generally also increase the accuracy of the model. More modern approaches make use of deep convolutional neural networks providing better results due to advantages such as preprocessing by extracting hidden features.

The study compares the performance of three different algorithms (Nearest Class Centroid, Nearest Subclass Classifier and Nearest Neighbor), analyzing the accuracy and the execution time of both training the model and classifying the test-set. The analysis has been conducted using two different standard datasets, MNIST and ORL, studying how the same algorithm performed on different database and how the number of training datapoints and features can impact the success rate and execution time. For better examinate the influence of the number of features, various Principal Component Analysis (PCA) have been applied, reducing the number of dimensions while keeping more information as possible.

The paper is organized as follow: in Section II are described the two datasets, how the classification algorithms work, what is a PCA and how the MATLAB code is structured; Section III shows the results of the study, analyzing the accuracy and the execution time; Section IV summarized the conclusion of the paper.

II. METHODS

A. Dataset Description

Two separate databases have been used, both formed by images, in order to better compare the classification schemes.

The first dataset is the MNIST (Mixed National Institute of Standard and Technology) database, commonly used for training and testing image processing systems. It contains a large number of black and white handwritten digits which were normalized to fit into a 28x28 pixel bounding box. Though, each datapoint has a dimension of 784 features, one for each pixel. The dataset has been split in 60’000 training samples and 10’000 testing images. In *figure 1* is shown a 2D representation of the MNIST database.

The second dataset is the ORL (Olivetti Research Laboratory) database, composed of 40 people faces, 10 black and white pictures per person. The 400 images have been divided in 280 training sample and 120 testing sample. Each

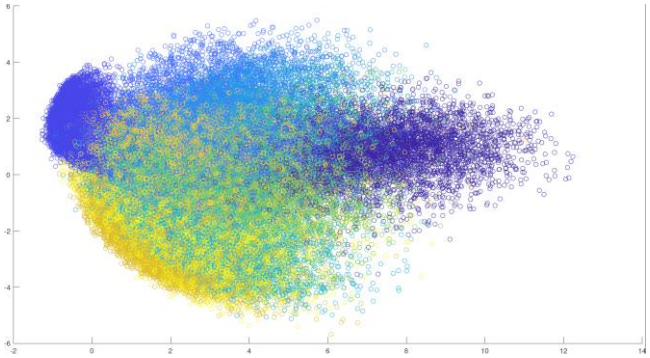

Figure 1: PCA in 2D of MNIST database

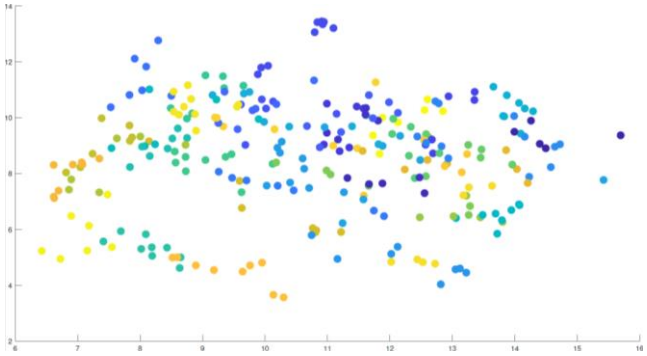

Figure 2: PCA in 2D of ORL database

datapoint has dimension of 1'200 since the images are 40x30 pixels each. In *figure 2* is shown a 2D representation of the ORL database.

We can see how these two databases differ in number of samples (70'000 / 400), number of dimensions (784 / 1'200) and number of classes (10 / 40).

B. Classification Schemes

The three classification schemes compared in the research are all based on cluster.

Nearest Class Centroid. The NCC classifier represents each class with a mean class vector computed starting from the training samples. Each mean class vector is called *centroid* of the class. Given the centroid μ_k of class *k*, the point x_* is classified in the smallest distance class:

$$d(x_*, \mu_k) = \|x_* - \mu_k\|_2^2$$

Though, during the training the algorithm finds the centroids for each class and during the prediction of a new sample scan all the centroids for finding the closest. This operation could be naively done in time:

$$O(DNK) \qquad \text{(II.B.1)}$$

Where *D* is the number of dimensions, *N* the number of points to classify and *K* the number of classes.

Nearest Subclass Classifier. For each class, the NSC searches several subclasses applying a clustering algorithm, in this study was used K-Means. The number of subclasses is a parameter of the algorithm and it has been set to 2, 3 and 5 during the evaluation, examining three different NSC. Each subclass *m* of class *k* is represented by a mean vector μ_{km}, and the new point x_* is categorized to the class corresponding to the smallest distance:

$$d(x_*, \mu_{km}) = \|x_* - \mu_{km}\|_2^2$$

During the fitting, the algorithm repeatedly applies K-Means to each class finding all the sub-centroids. Predicted a new sample means scanning all the centroids of the subclasses and find the closest, this could be done in a time complexity of:

$$O(DNKQ) \qquad \text{(II.B.2)}$$

where *D* is the number of dimensions, *N* the number of points to classify, *K* the number of classes and *Q* the number of subclasses for each class.

Nearest Neighbor. The NN classifier is an extreme case of NSC, in which the number of subclasses per class is equal to the number of its samples. In this algorithm, though, the fitting time is zero while the prediction requires to scan all the training samples having a complexity of:

$$O(DMN) \qquad \text{(II.B.3)}$$

where, again, *D* is the number of dimensions, *N* the number of new points to classify and *M* the number of training samples.

Simply looking to the theoretical time complexity of the predictions and to the description of the training, it is possible to have an idea of the performance of each scheme.

C. PCA

The Principal Component Analysis (PCA) aims to reduce the number of dimensions keeping more information as possible. The analysis looks for an orthogonal transformation in which the dimensions are sorted based on their variance, starting with the largest possible.

First the scatter matrix S_T of the training samples is calculated:

$$S_T = \sum_{i=1}^{N} (x_i - \mu)^T (x_i - \mu)$$

where μ is the mean of all samples and x_i is one of the *N* samples.

Then, the eigen-vectors of S_T are found and sorted according to the descending order of the corresponding eigen-values. The matrix *W* formed by these eigen-vectors is

the transformation matrix and each datapoint is pre-multiplied for its transposed:

$$y_i = W^T x_i$$

According to how many eigen-vectors are kept, it is possible to define different transformations, mapping the data from $\mathbb{R}^D$ to $\mathbb{R}^d$, with $d < D$.

Figure 1 and *2* shown a PCA applied respectively to the MNIST and the ORL databases, keeping only the 2 dimensions with the highest variance. From these images, it is possible to notice how samples of the same class naturally form clusters.

D. Code Structure

The code has been developed in MATLAB, due to its native matrix approach. For each classification scheme, a class [1] [2] [3] has been defined with the following methods:

```
[obj, time] fit(training_data, training_labels)
[accuracy, time] score(test_data, test_labels)
```

The first method trains the model, returning the object of the trained classifier and the time spent throughout the fitting. The second method computes the accuracy of the model by using the testing data and labels passed as parameters. It returns the accuracy and the time spent for the computation. Even if the NN classifier does not require a proper *fit* method, it has been left for coherency with the other classification schemes.

K-Means and PCA are implemented as two MATLAB functions [4] [5]. In the first one [4], the stopping conditions can be configurated, the default values are 150 iterations or a variation in the centroids position less than 10^{-4} (the function stops at the first condition met). The second one [5] returns the transformation matrix which should be applied both to training samples and to testing data.

Where possible, instead of looping for each sample, a matrix computation has been performed in order to speed up the calculation. This means that the complexities (II.B.1), (II.B.2) and (II.B.3) are only theoretical since the matrix operations of MATLAB are more optimized, nevertheless they are still a valid upper bound.

The *main* script [6] is structure as follow:

First the path dependences are imported to correctly execute the whole code. Then, the data are loaded from the two databases. If MNIST was already split into training and testing set, ORL has been split in the code leaving 70% of the data for training and the remaining for testing. This second database must be shuffled since all the datapoints are sorted according to their labels. Due to the random shuffle, training samples are always different and performance at each execution could change. After that, several PCA are performed and the transformed data saved in MATLAB Cells. The dimensions kept during the PCA are 2, 1%, 2%

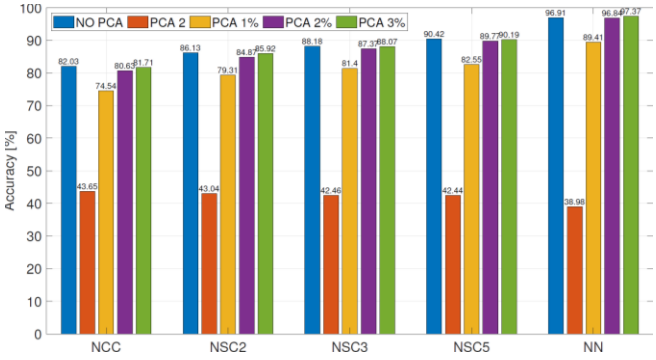

Figure 3: Accuracy in MNIST dataset

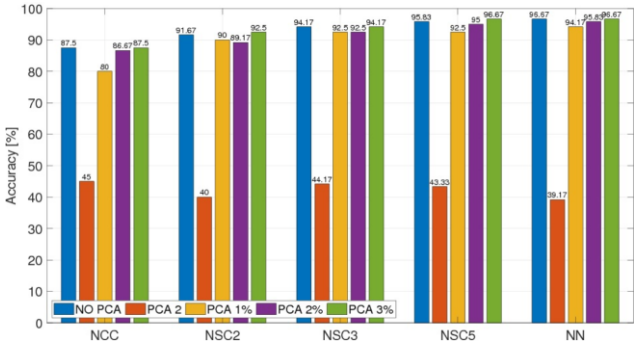

Figure 4: Accuracy in ORL dataset

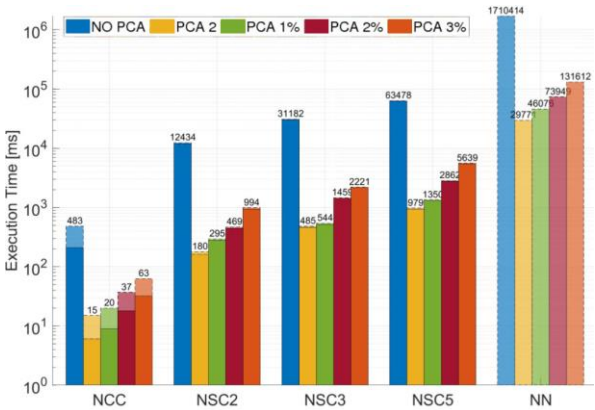

Figure 5: Execution time in MNIST dataset. Y axe has a logarithmic scale. The top and lighter zone of each bar represent the *score* time, while the darker zone the *fit* time

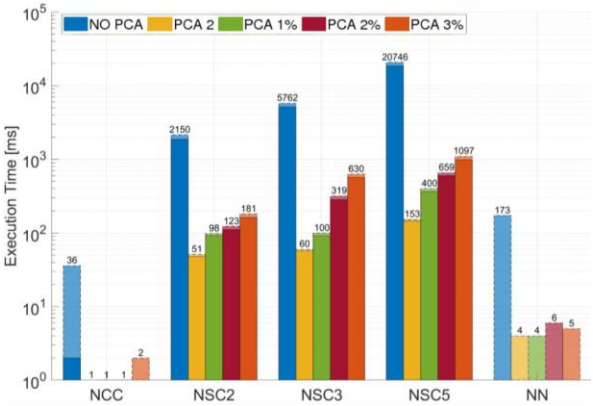

Figure 6: Execution time in ORL dataset. Y axe has a logarithmic scale. The top and lighter zone of each bar represent the *score* time, while the darker zone the *fit* time

and 3% of the original dimensions.

Now all the classification schemes are executed saving *accuracy*, *fit time*, and *score time* for each of them. The NSC has been performed three times with number of subclasses equals to 2, 3 and 5.

Lastly, some plots are printed and saved using the data collected during the analysis.

III. Results

A. Accuracy

Looking at *figures 3* and *4*, we can notice how, in both the databases, the accuracy increases as centroids per class increase.

In the NCC, with only one centroid per class, the accuracy is the lowest between all the algorithms while in NSC with 5 sub-classes and in NN classifier, the accuracy is the best overall, touching peaks of 96.91%. The score of the ORL database is often better than the MNIST one, even though it has less samples. Since ORL contains faces, data are more complex and various than the digits in MNIST where most of the pixels are very light with only few dark ones in correspondence of the pen sign. The higher number of features and the larger variance bring an advantage to the ORL's samples.

We will now focus on the PCA results. Reducing the dimensions at only two, we can see a drastic drop in accuracy, falling below 50% in all results. It is obviously not enough; the new two features do not contain sufficient information to distinguish the samples from the various classes. Rising the number of dimensions kept, the PCA accuracy tends to reach the one without the data transformation. With only the 3% of the original dimensions, i.e. 47 in MNIST and 71 in ORL, the algorithms could also perform a higher score than the original one, all for the benefit of the computational time.

B. Execution Time

The code was run on a Windows 10 portable machine powered by a sixth generation Intel Core i5. The execution time is therefore bound to the machine's calculation power.

We recorded the time for both the fitting phase and the scoring phase, the result can be seen in *figures 5* and *6*. For each bar in the graphs, the first and darker zone represents the time due to the fit method, while the second and lighter zone due to the score. Please notice the logarithmic scale, introduced for better display small values next to high one.

As anticipated, the fit time for the NN classifier is zero, since it must only save all the training samples and labels. Nevertheless, the score time is the highest since it loops for each training and testing data and we could wait up to 30 minutes in the MNIST case having a total of 70'000 datapoints. In the NSC, the time for training dominates the prediction time due to the fact of applying the clustering algorithm. Indeed, the more the subclasses, the higher is the time spent during the fitting. Once trained the model, NSC can perform a fast prediction over the new samples, becoming preferable to NN. NCC is always the quickest, having a fit me negligible and the shortest score time of all. This because the fit method must only compute the mean vector for each class and the score method loops only for each class for a total of *K* iteration (II.B.1), that are fewer both than the *M* testing samples - as in NN (II.B.3), and the total *QK* subclasses - as in NSC (II.B.2).

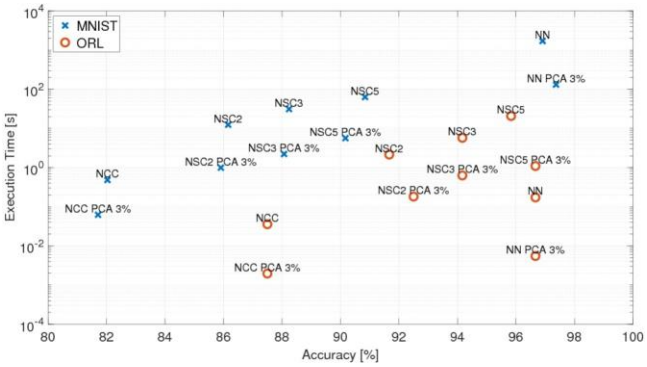

Figure 7: Comparison between all the databases and all the algorithm. Y axe is in logarithmic scale

If we consider the data transformed by the PCA, we can see a huge increase of speed in every classification scheme. Dimension reduction brings an improvement in every vector and matrix calculation, which are very common in every algorithm.

Directly compering MNIST and ORL databases is even clearer how the number of samples impact the execution time: all the algorithms applied to the second database are faster.

Finally, *figure 7* shows a direct comparison between all algorithms and data. The best performances can be found in the lower left corner, with higher accuracy and lower execution time. From this picture we can deduct that the algorithms performed better if applied to ORL dataset since all the points on the graph are lower and righter than MNIST ones.

IV. Conclusion

Image Classification relies on the relationship between the data and the classes into which they are classified, and we tried to exploit this relationship using cluster classification schemes. We have compered three different algorithms implemented in MATLAB applied to two different databases. NCC results to be faster classification schemes, but also the one with the worst accuracy. NSC spent the most time during the training, resulting faster in the prediction of new samples compered to NN and with a good accuracy. NN has a fit time equals to zero but a prediction time quite long, nevertheless it results the best in accuracy. In general, as the number of cluster/sub-cluster per class increase, we have noticed an increasing also in the accuracy of classification

with a downside on the execution time. The PCA can greatly improve the execution time of every algorithm and keeping just the 3% of the original dimension we can achieve the same accuracy as the one with the original data.

CODE REFERENCES

[1] D. Galli, *NCC.m*, Code/Classifiers/
[2] D. Galli, *NSC.m*, Code/Classifiers/
[3] D. Galli, *NNC.m*, Code/Classifiers/
[4] D. Galli, *kmeans.m*, Code/Functions/
[5] D. Galli, *PCA.m*, Code/Functions/
[6] D. Galli, *main.m*, Code/