K-NEAREST NEIGHBOUR CLASSIFIER

Student Gianluca Galvagni, S5521188, Università degli Studi di Genova

***Abstract*—How third assignment, we talked about k-nearest neighbour classifier and the evaluation of classifiers. KNN or k-NN is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. Our goal for this assignment was created a code which can make by two different dates set the classification of them and then the code has to calculate the error rate, the difference with different k values and the evaluation of those classifications.**

***Index Terms*—K-Nearest Neighbour, plurality voting, majority vote.**

I. INTRODUCTION

CLASSIFICATIONis a model for recognition. With a given input, the role of the classifier is to categorize this input in one of the suitable class. For example, I have a patient record as input and I want to put the patient in one of the categories healthy/at risk/ill.
The k-Nearest Neighbour classifier is a solution for this problem. Differently from others (for example the Naive Bayes Classifier) this is a non-parametric model. This means that I do not explicitly implement a model with parameters, but I directly build a discrimination rule from data. So, there is not a training-phase. In this work I implement in Matlab the k-Nearest Neighbour Classifier, and test it on different data with different values of k.

II. THE KNN CLASSIFIER

In this first part, I want to build a k-Nearest Neighbour classifier. Given a query point $\overline{x}$ I want to classify it. So I build a rule (the classifier) $y()$ that receives the query point $\overline{x}$ and outputs a class (the category) $w = y(\overline{x})$. The idea is to do a mechanism of voting for the choice of the class; the majority will decide the right class. I have the training set X:

$$X = \begin{bmatrix} x_1^T \\ \cdot \\ \cdot \\ \cdot \\ x_n^T \end{bmatrix}$$

Where each x_l^T is a row with $d+1$ columns which is a training point with a known class $x_{l,d+1}$.
The algorithm performs the norm between each training point x_l^T and the *query* point $\overline{x}$:

$$N = \{\|x_1 - \overline{x}\|, ..., \|x_n - \overline{x}\|\}$$

If the columns of $\overline{x}$ are greater than columns of x_{lT} (minus the $d+1$ that is the class) the additional columns are discarded at the beginning and so not considered, in order to perform correctly the norm.

The values of N tell us the *distance* of the query point from each training points, so the smallest k values refer to the k nearest neighbours.
Thus, I extract the smallest k norms from N.

$$\{N_1, ..., N_k\} = min - k\{N\}$$

And let the corresponding training points $x_{N_1,...,}x_{N_k}$ vote for the output class w:

$$w = mode\left\{x_{N_{1,d+1},...,}x_{N_{k,d+1}}\right\}$$

Where the mode is an operator that extracts the most frequent value. Many variants exist (for example using an approximated, but faster, formula to calculate the distance) but they are not covered in this work.

III. PROS AND CONS OF KNN CLASSIFIER

Having no training phase, the cost for it is $O(1)$: just for storing the training set. But, what I gain in learning phase, I pay in classification complexity: $O(nd)$. In particular, the most critical phase is the computation of the min-k: I have to scroll the vector of the norms k times. For not small values of k it is more convenient to sort the vector before, but this require time, too (O($n\,log(n)$)). This kind of classifier is theoretical guaranteed: for $n \to \infty$, the error rate is $\leq\ 2\ \times\ BayesError$, where Bayes Error is the best theoretically achievable error.
However, in practice, errors can be consistent if the decision region have "jagged" borders. This is the case when the training points are near (so, very far to be *linearly separable*).

IV. LAB ACTIVITY

In this assignment I have used Matlab R2022b to make my program. My work consists in the development of a k-Nearest-Neighbour classifier on a MNIST data set.So the goal is to given a classification of the data represent 70 000 handwritten digits in 28x28 grey scale images.

1) *TASK 1:* ***Obtain a data set.***
   The data that I analyzed is a MNIST data set; the data represent 70 000 handwritten digits in 28x28 greyscale images, already split into a 60K-image training set and a 10K-image test set, and are a standard benchmark for machine learning tasks.
   The task 1 loads the data specifying if i want load the training set and the respective labels, or the test set and its labels.
2) *TASK 2:* ***Build a KNN classifier.***
   In this task i implemented the k-Nearest-Neighbour classifier. This was done applying the theory, in particular the task calculates the norms between all the training points and a query point. After, I must extract from this

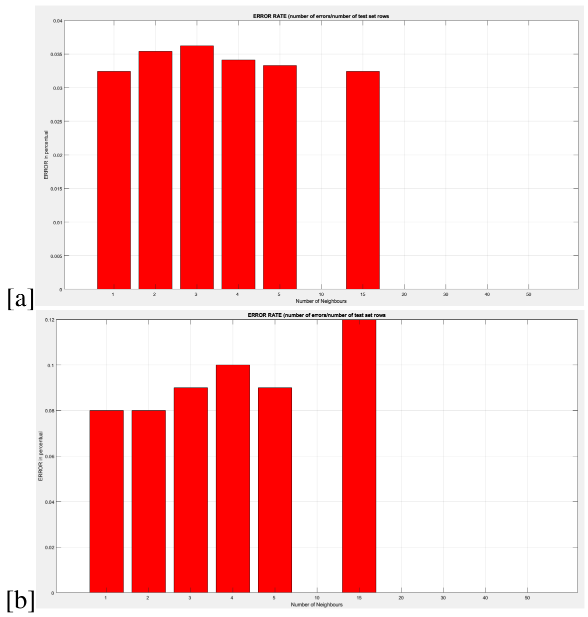


Fig. 1. ERROR RATE, number of error / number of test set rows.
(a) It represents all dates in the the train set e test set.
(b) It represents only few dates.

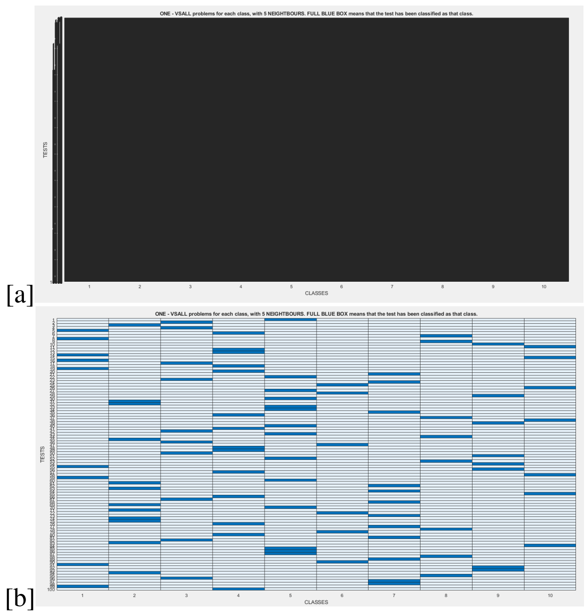


Fig. 2. CLASSIFICATION IN THE CLASSES.
(a) It represents all dates in the the train set e test set.
(b) It represents only few dates.

norms the k smallest ones: This was done ordering the norms vector and taking the first k values. With this method, I improve the computational time with respect to research k times the minimum.
At the end, if the test set has numbers of columns equal or higher than the training set, the $d+1$ column (where d is the column of the training set) is used to calculate the error rate between the obtained classifications and the real results.

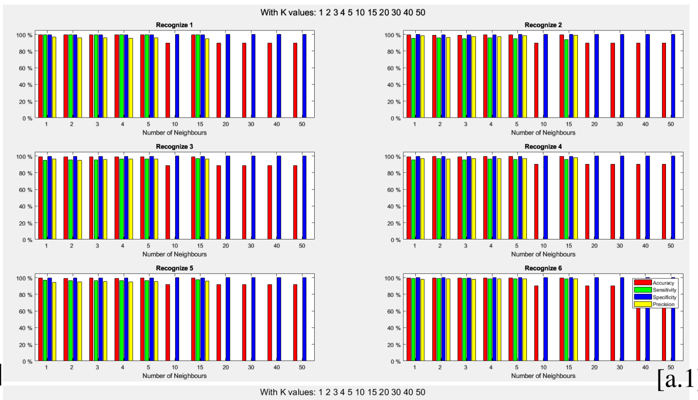
3) *TASK 3:* ***Test the kNN classifiers.***
In this task, we computed the accuracy on the test set using the MNIST character recognition data. I checked the accuracy on 10 tasks: each digit *vs* the remaining 9. Also I controlled the correctness of the classifier for several values of k using the rule that k should not be divisible by the number of classes.
4) ***ADDENDUM:***
In the last part, I do some statistics over the last repeated experiments. For each of the experiments above, I computed a confusion matrix, and then from it classification quality indexes. I made an indication of typical value (*an average*) and an appropriate measure of spread (*a standard deviation*. And then, I plotted all results.

V. Conclusion

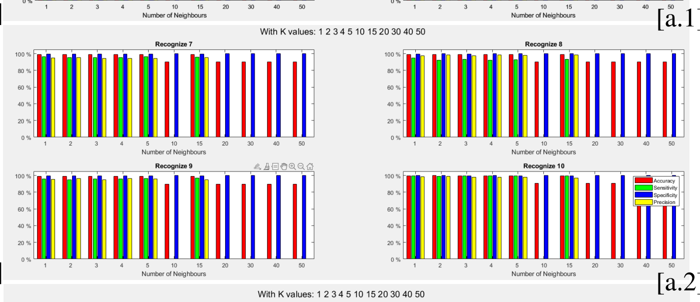
At the end of this third assignment, there are some points of view to analyze.

- How it can see, k should not be divisible by the number of classes and i did not calculate the error rate and the confusion matrix for those values.
- In the Fig.1, there are a big difference between the big training set and the smaller. This was what I was waiting like result.
- In the Fig.2a, I can not see anything because the number of the *TEST* are too big and the graph can not show the result but it is like the Fig.2b, with more tests.
- In the Fig.3, there are all kind of possibility to evaluation of classifier. The results are similar together but, of course the biggest training set is more sharp than the small training set. And for the *addendum* (Fig.4) the results have the same behavior.

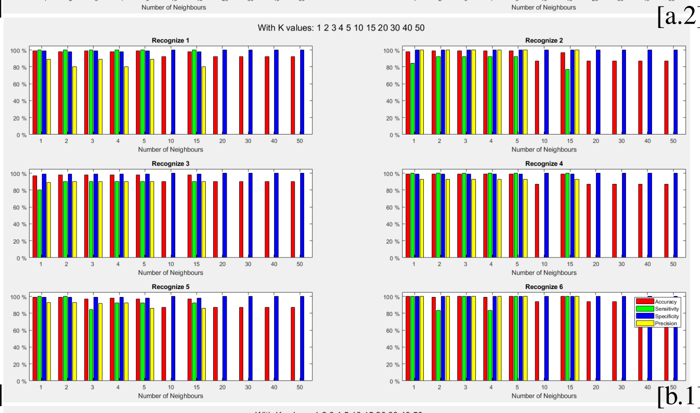
I reported only two of the possible running. Up to now, I assume that results acceptable for this assignment.



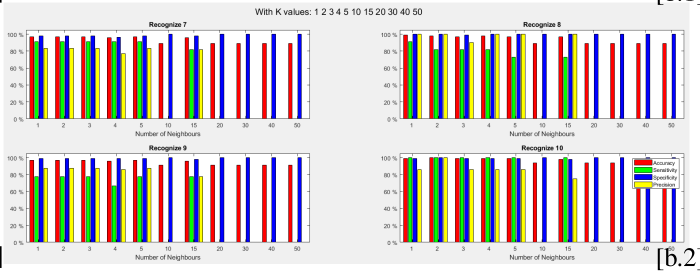
[a.1]



[a.2]

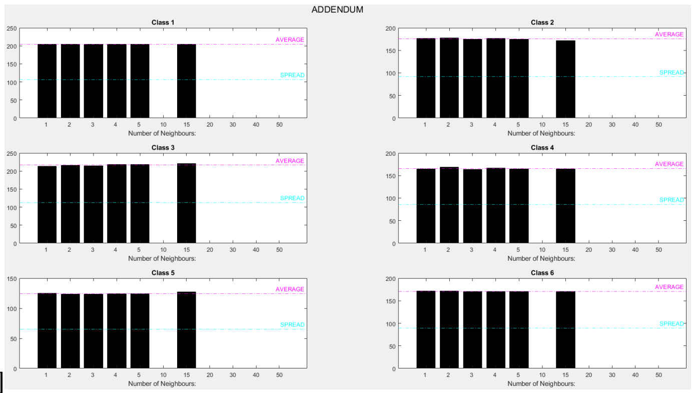


[b.1]

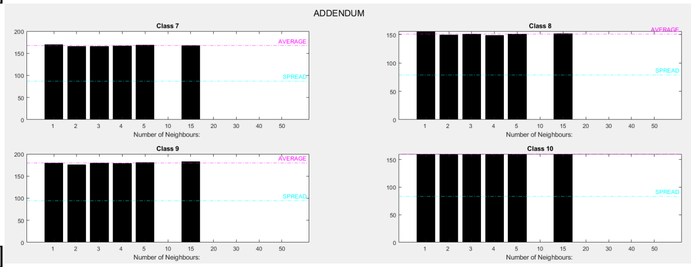


[b.2]

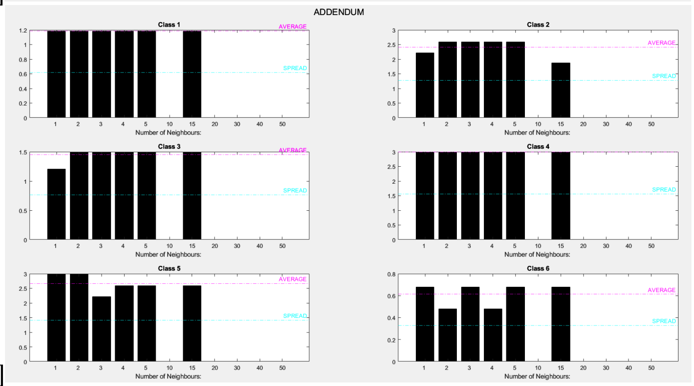
Fig. 3. EVALUATION OF CLASSIFIER.
(a.1-a.2) It represents all dates in the the train set e test set.
(b.1-b.2) It represents only few dates.



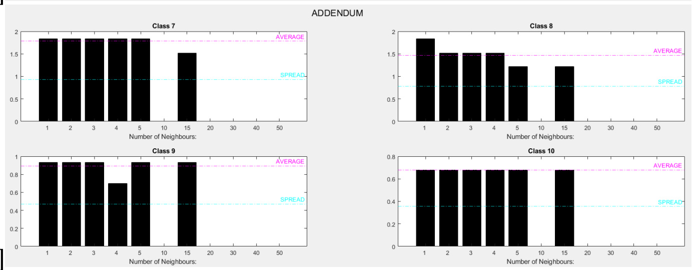
[a.1]



[a.2]



[b.1]



[b.2]

Fig. 4. ADDENDUM.
(a.1-a.2) It represents all dates in the the train set e test set.
(b.1-b.2) It represents only few dates.