kNN CLASSIFIER
Assignment 3 of Machine Learning I, A.Y. 2021/2022

Marco Macchia
DIBRIS - Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi
Università Degli Studi di Genova

This report describes the third assignment of Machine Learning I, what were the requirements and what are the obtained results.

I. Introduction

CLASSIFICATION is the process of recognize the data input and associate it to a *class*. One of the most common example regards the classification of the e-mails, where the classifier tries to understand if an incoming mail can be categorized as a *safe* email or as a *spam*.

There are many types of classifier, e.g. the *naive Bayes classifier*. Another type of classifier is the **k-Nearest Neighbour classifier**. One of the main features of the kNN classifier is the fact that it does not require any training session during its process, because it computes the *discriminant function* $g_i()$ right during the classification phase.

II. Theory of kNN

A. Mathematical theory

Given an input point $\overline{\mathbf{x}}$ called *query*, the goal is to classify this point using all the other points that surrounds it. The classifier then gives the predicted class ω as output, such that $\omega = y(\overline{\mathbf{x}})$, where $y()$ is the rule of the kNN.

The decision rule is based on the concept of the **majority voting**, meaning that the $\overline{\mathbf{x}}$ will be classified with the class ω_1 only if the majority of the k-closest objects around him belongs to the class ω_1.

Let's define X as the training set given as input to the kNN

$$X = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_n \end{pmatrix} \tag{1}$$

where $\mathbf{x}_i$ is a single observation of the training set. The k-Nearest Neighbour classifier computes the *distance*(norm) between each observation $\mathbf{x}_i$ and the query point $\overline{\mathbf{x}}$, and stores it in a row vector $\mathbf{N}$

$$\mathbf{N} = (||\mathbf{x}_1 - \overline{\mathbf{x}}||, ||\mathbf{x}_2 - \overline{\mathbf{x}}||, ..., ||\mathbf{x}_n - \overline{\mathbf{x}}||) \tag{2}$$

where each value stored inside $\mathbf{N}$ tells how much the query point is similar to the considered observation.

The kNN classifier than takes into account the first k points that are closest to $\mathbf{x}_i$ (i.e. the k points that are most similar to the query one)

$$\{n_1, ...n_k\} = \text{top-}k||\mathbf{x}_i - \overline{\mathbf{x}}|| \tag{3}$$

The class given to $\mathbf{x}_i$ is then the class that appears the most:

$$\omega = mode\{t_{n_1}, ..., t_{n_k}\} \tag{4}$$

where t_{n_1} is the class of the class 1.

B. Properties of Nearest Neighbour

Since the classifier has no *training session*, the complexity of learning is $O(1)$, because it just has to store the training set.

However, the classification complexity is instead much higher, and it corresponds to $O(nd)$. The *critical* section of the classifier is the search for the minimal value inside $\mathbf{N}$ that has to be done for each observation of the data set that we want to classify.

NN classifiers are *theoretical guaranteed*, which means that

$$when\ \ n \to \infty \qquad errorRate \leq 2BayesError$$

where Bayes Error is the best theoretically achievable error.

However, in the case that the training set is made by observation that are too close to each other, NN classifiers may *overfit*, i.e. decision regions may have jagged borders.

III. The Assignment

The given assignment consist of three tasks:
- **Task 1:** Obtain a data set
- **Task 2:** Build a kNN classifier
- **Task 3:** Test the kNN classifier

A. Task 1: Obtain a data set

This assignment takes into account the *MNIST* set, a standard benchmark for machine learning tasks. The set is made of 70.000 observations (60.000 for the training set and 10.000 for the test set), 784 attributes and 10 classes.

Each observation represents an handwritten digit(from 0 to 9) stored in a 28x28px greyscale image. When the training set and the data set are loaded, the images are stores in a 784-rows vector.

Since the two sets are very big, two random subsets are selected inside the main function.

B. Task 2: Build a kNN classifier

In the second tasks, we have to actually implement a k-Nearest Neighbour classifier. First it has to check the correctness of the input variables, then it has to classify the test set according to the kNN theory described with formulas 2,3 and 4, and return the classification obtained.

In particular, the classifier, for each observation in the test set, computes **N** using the *euclidean norm*. In order to make the computational steps easier, the function *vecnorm()* was used, because it is able to compute the norm directly on the single rows, without having to manipulate the input data.

For each observation then the kNN classifier finds the k-minimum values inside of **N**, and stores their indexes. The classification given to the *query point* $\bar{\mathbf{x}}$ is then the mode of the classes that are stored at the minimum indexes.

Moreover, if the test set ground truth vector is given, after assigning at each *query point* a class, the classifier should also compute the total error rate, and return it to the main function.

C. Task 3: Test the kNN classifier

In task 3 it is required to run the kNN classifier using different values of k. The values used are:

$$\mathbf{K} = [1, 2, 3, 4, 5, 9, 15, 19, 29, 39, 49]$$

Note that the values of k that are divisible by the number of classes are discarded, in order to avoid ties. Task three also requires to compare the accuracy of the classifier on the different classes (i.e. digits), checking each digit *vs* the remaining nine.

IV. Results

As the training set and the data set are quite big, two subsets are randomly selected in order to speed up the process. In particular for this simulation the subsets has the 3% of the total size of the sets, so that the training set has 1800 different observations and the test set has 300 observations.

This procedure is called *condensed NN classification*, where the sets are made of $c < n$ observations rather than all n data. The complexity order then becomes $O(cd)$.

Moreover, in every execution of the classifier the test set ground truth is given, in order to always compute the error rate, useful for the comparisons.

Notice that in this simulation the class 10 corresponds to the digit 0.

As shown in the image 1, the error rate increases as the value of k gets higher. The best value of k should then be lower then 10, in order to avoid taking considering some wrong neighbours.

The figure 2 shows instead a comparison between the mean error rates computed for each different class, using different values of k. It is clearly visible that the easiest class to recognize is 1 (with an error rate lower than 0.05), while the hardest class to recognize is 3 (with an error rate of about 0.5, meaning that one classification every two is wrong).

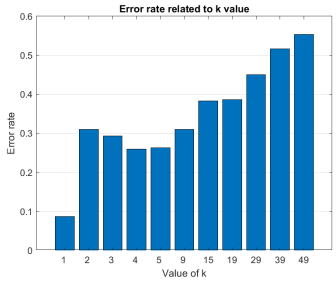

Fig. 1: Error rate with different values of k

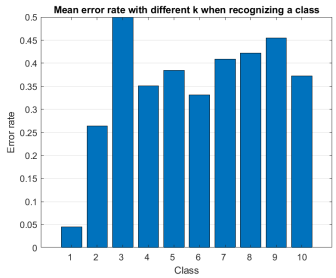

Fig. 2: Mean error rate among classes

Figure 4 and figure 5 shows instead the performance of the kNN with respect to the single classes. In particular, with the given subset, the figure 4 highlights that generally speaking, using $k = 1$ is the best choice. The classifications has good performance also with $k = 4$ and $k = 5$, while it the performance lowers as the value of k increases. Note that the worst classifications are given with $k = 49$.

Figure 5 considers the recognition rate of a class with different values of k. with this representation it is clear that, also with higher values of k, the classifiers is able to easily understand whether the observation corresponds to a 1.

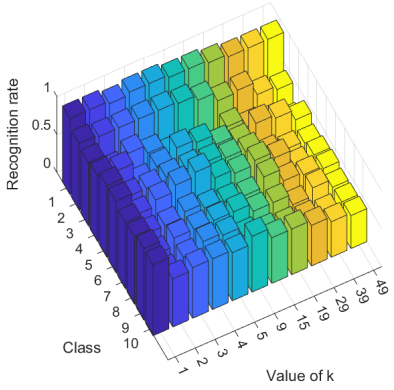

Fig. 3: 3d representation of recognition rate

Finally, figure 3 is a 3D representation of what has been described in figures 4 and 5.

Generally speaking, the classifier did his best understanding the class 1 with $k = 1$, $k = 2$, $k = 4$ and $k = 5$, while were practically unable to understand a 3 with $k = 15$.

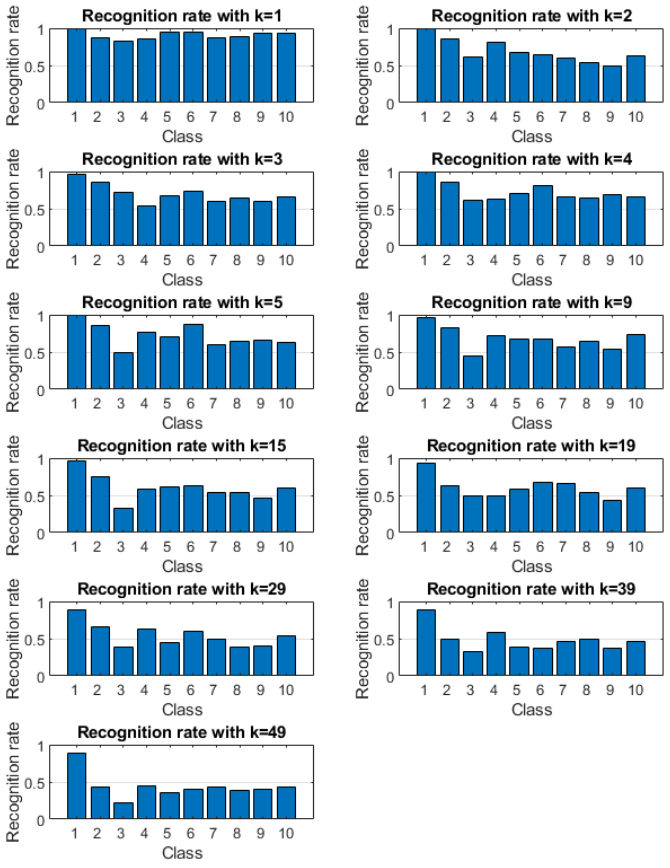

Fig. 4: Recognition rate (with different values of k)

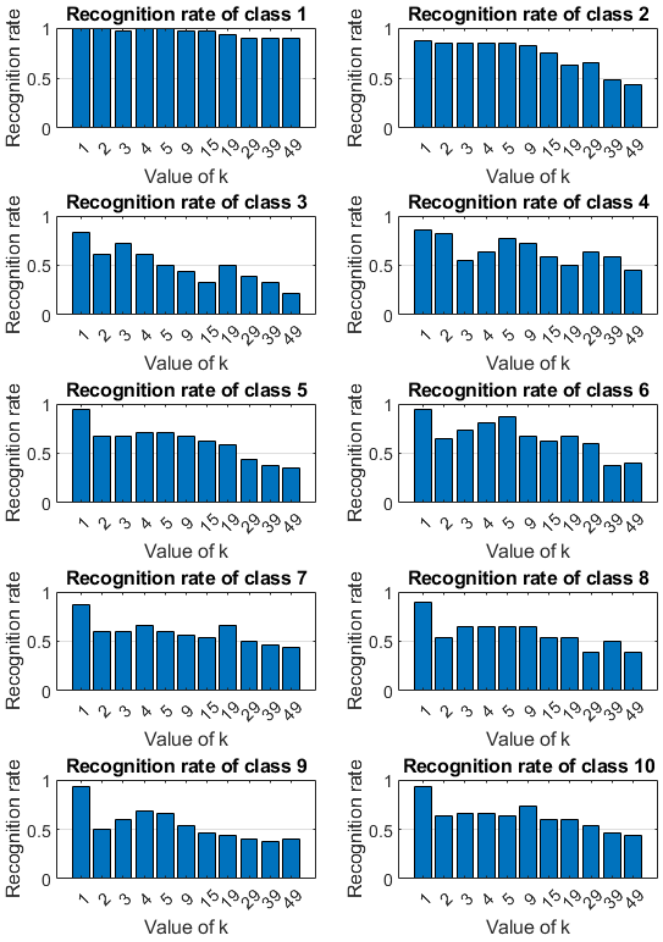

Fig. 5: Recognition rate for each value of k (on the classes)