Machine Learning - Robotics Engineering
Report Assignment 3
kNN classifier

Francesca Canale 4113133

25/11/2019

Abstract

This assignment is based on the k-nearest neighbour classifier. Classification is a model for recognition: with a given input, the role of the classier is to categorize this input in one of the suitable classes. Differently form other types of classifier the kNN is a non-parametric model: this means that we do not explicitly implement a model with parameters, but we directly build a discrimination rule from data. So, there is not a training phase. Due to this fact this algorithm has an high computational cost at the classification time and it gets significantly slower as the number of examples/variables increase.

1 Introduction

The goal of this lab assignment was to build a k-nearest neighbors classifier and to test its accuracy on a given data set trying with different values of k.
This was implemented with three tasks:

- **Task 1:** Obtain a data set;
- **Task 2:** Build a kNN classier;
- **Task 3:** Test the kNN classier.

2 Task 1

We have been given a single data set to work with. It is the MNIST database of handwritten digits that has a training set of 60000 examples and a test set of 10000 examples. Each example of both the training and the test set is a 28x28 pixels image containing an handwritten number form 0 to 9. In Fig. (1) there are some examples of images taken from the training set. To load the data set on MATLAB we have been given a function *loadMNIST()* that for each set returns:

- a matrix that has as many rows as the number of examples (so 60000 rows for the training set and 10000 rows for the test one) and 784 columns that represent the value of each single pixel of the images;
- a column, as long as the number of examples, that contains the labels from 1 to 10 which represent which digit is shown in the image (10 corresponds to the number 0);

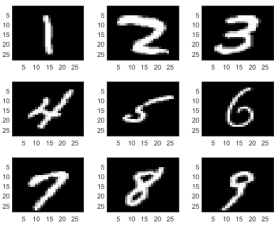

Figure 1: Examples of images of the training set

3 K-nearest neighbors classifier

KNN is a non-parametric technique and one of the most widely used supervised learning approach. The kNN algorithm assumes that similar things exist in close proximity: in other words, similar things are near to each other [1]. Given a set of n training examples, upon receiving a new instance to predict, the kNN classifier will identify k nearest neighboring training examples of the new instance and then assign to the new instance the class label holding by the most number of neighbors [2]. The algorithm works as follows:

Algorithm 1 kNN Algorithm

Require: A training set X = $\{x_1,..., x_l,..., x_n\}$
Require: A "query" point $\overline{x}$
Require: Value of k
Output: Classification y for the query point
1: $\{n_1, ..., n_k\}$ = top-k $||x_l - \overline{x}||$
2: $y = mode\{t_{n_1}, ..., t_{n_k}\}$

3.1 Task 2

In order to implement the kNN algorithm, I created a MATLAB function *kNN_classifier()* that takes in input a training set, a test set, a value of k and optionally the test set ground truth. The function returns the classification for each observation in the test set and, if the fourth input is available, the error rate.
Firstly, this function checks if the test set has one column less than the training one (the class column) and it checks if the value of k is good (it must be greater than zero, lower than the number of observations in the training set and it is recommended that it is not a multiple of the number of classes).
Secondly, it starts implementing the algorithm described in Algorithm [1] considering as "query" points every observation in the test set and using some already available MATLAB functions:

- $[D, I]$ = pdist2(X, Y, 'euclidean', 'Smallest', k) that implements point (1) of the alorithm. It computes the euclidean distance D and returns the k smallest pairwise distances to observations in X for each observation in Y in ascending order. I contains the indices of the observations in X corresponding to the distances in D [3] and it is used to determine $\{t_{n_1}, ..., t_{n_k}\}$.
- M = mode(A) that implements point (2) of the algorithm. It returns the most frequent value of A [4].

If the fourth input is available, the error rate is computed as the number of times the classification obtained from the algorithm is different form the ground truth, divided for the number of observations in the test set.

4 Task 3

For this lab assignment it was required to test the classifier computing the accuracy on the test set using the MNIST character recognition data. We checked the accuracy on 10 tasks: each digit vs the remaining 9, so identifying whether an observation has a class value or not, considering also several values of k. In order to check the accuracy I implemented a MATLAB function *accuracy_plotting()* that, taking in input the error rate computed by *kNN_classifier()*, computes and plots the accuracy.

5 Results

Below are reported the error rate obtained on task 2 (Fig. (2)) and the accuracy of task 3 (Fig. (4)). The values of k chosen are those included between 1 and 150, not taking into account those that were multiples of the number of class (so multiples of 10 for the second task and multiples of 2 for the third) in order to avoid ties.

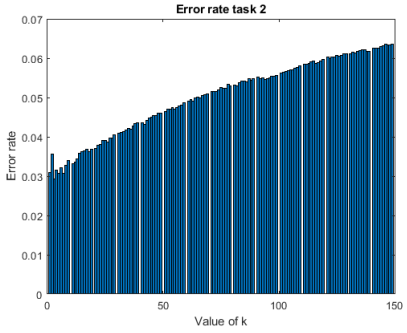

Figure 2: Task 2 Error rate

In the bar graphs we can notice that the best values of k (number of neighbours) for both tasks are the smallest ones. Indeed the error rate increases (and so the accuracy decreases) for values of k too big: this is because too many neighbours are taken into consideration and so also "not so near" values vote.
The accuracy for each digit of task 3 is very high (in Fig. (4) are reported only values of the accuracy between 97% and 100%) this is because recognizing whether a handwritten number is a specific digit or not is an easier task than having to identify which digit it actually is.
In Fig. (3) I report two examples of cases in which the classifier makes the wrong classification: in both the examples t is the true class of that test sample while y is the classification obtained and the value of k is 3 (that was the one with lowest error rate in task 2).

Figure 3: Examples of wrong classification

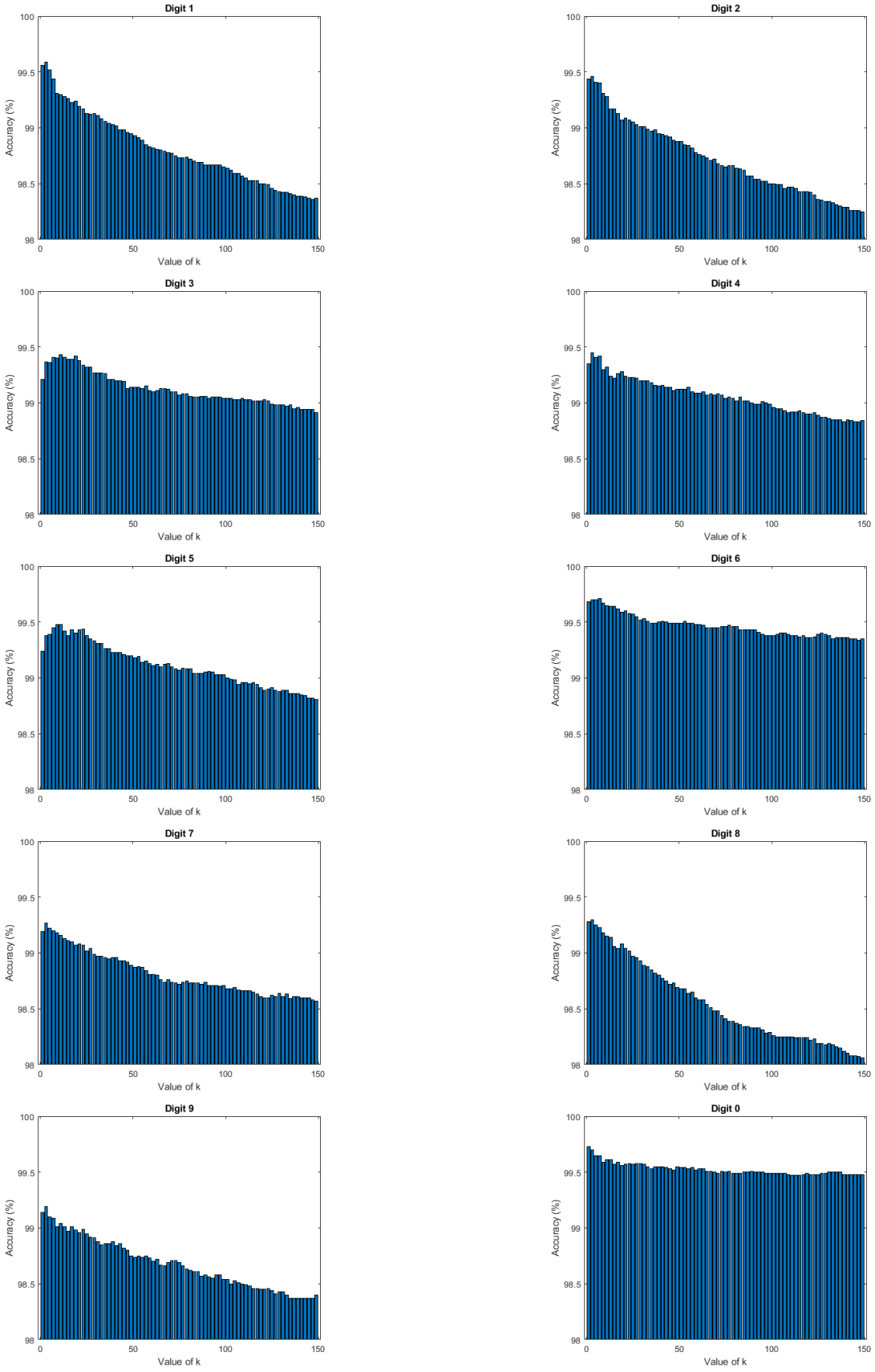

Figure 4: Accuracy for each class

6 Conclusion

The kNN classifier is simple, easy to implement and for the right choice of k provides results with quite low error rate. But the main drawback of kNN is its inefficiency for large scale and high dimensional data sets. The main reason of its drawback is its "lazy" learning algorithm natures and it is because it does not have a true learning phase and that results a high computational cost at the classification time [5].

References

[1] "Machine Learning Basics with the K-Nearest Neighbors Algorithm" - Towards Data Science - https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

[2] "Editing Training Data for kNN Classifiers with Neural Network Ensemble" - Yuan Jiang and Zhi-Hua Zhou

[3] "pdist2 - Pairwise distance between two sets of observations" - MathWorks Documentation - https://it.mathworks.com/help/stats/pdist2.html

[4] "mode - Most frequent values in array" - MathWorks Documentation - https://it.mathworks.com/help/matlab/ref/mode.html

[5] "A Review of Data Classification Using K-Nearest Neighbour Algorithm" - A. Kataria and M. D. Singh - International Journal of Emerging Technology and Advanced Engineering