Classifying images with K-nn Method

Name Surname: Beria Ayşenur Can
Student Id: 090200705
CRN: 11512
Instructor: Emre Onur Kahya

1. Homework :

a) Create a dataset by first converting colored ones to grayscale images with more than 500 images for each class. (Please don't choose cats and dogs)
b) Write a knn classifier only by using numpy (no scikit-learn or other libraries which does it for you automatically)
c) Train it with 90 of your data and test it with 10 of it and see what the accuracy is.
d) Draw the training and testing curve as a function of the number of epochs..

2. Aim

In this example we will be applying machine learning and we will be using Knn method to be able to predict images whether it is a cat or a rabbit.
In addition, we observe how accurately the knn method works..

3. How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:
Step-1: Select the number K of the neighbors
Step-2: Calculate the Euclidean distance of K number of neighbors
Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.
Step-4: Among these k neighbors, count the number of the data points in each category.
Step-5: Assign the new data points to that category for which the number of

the neighbor is maximum.
Step-6: Our model is ready.

4. A close look at the structure of KNN

Suppose we have:
-a dataset D,
-a defined distance metric that we'll be using to measure the distance between the set of observations,
-and an integer K representing the minimum number of near neighbors we should consider to establish proximity.
In order to predict the output y for a new observation X, will follow these steps:
1.Calculate the total distances between the X observable and all the data points.
2.Retain the K observations that constitute the smaller distances to the observable point X.
3.With the y outputs taken from the K observations.
-apply the mean of the y deductions if it's a regression problem,
-use the mode of y deductions if it's a classification problem.
4.The final prediction will be the value calculated in step 3.
5.A detailed version of the algorithm can be found in pseudo-code:

Step 1 of 5

Nearest-neighbor algorithm

a) A pseudo code for the nearest neighbor algorithm is

ALGORITHM *Nearest-neighbor*$(D[1...n,1...n],s)$
//Input: A $n\times n$ distance matrix $D[1...n,1...n]$ and an index s of the starting city.
//Output: A list Path of the vertices containing the tour is obtained.
for $i \leftarrow 1$ to n do Visited $[i] \leftarrow$ false
Initialize the list Path with s
Visited $[s] \leftarrow$ true
Current $\leftarrow s$
for $i \leftarrow 2$ to n do
    Find the lowest element in row current and unmarked column j containing the element.
    Current $\leftarrow j$
    Visited$[j] \leftarrow$ true
    Add j to the end of list Path
Add s to the end of list Path
return Path

5. Data:

The dataset includes 2 different folders for train, test.Our dataset includes 800 images for each category (cats and rabbits).

Figure 1: cat

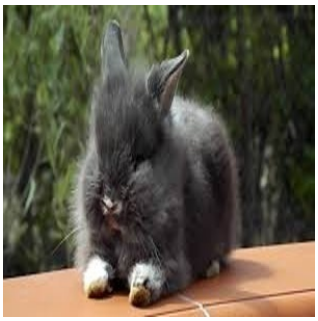

Figure 2: rabbit

6. Analysis :

The KNN algorithm can compete with the most accurate models because it makes highly accurate predictions. Therefore, you can use the KNN algorithm for applications that require high accuracy but that do not require a human-readable model.
The quality of the predictions depends on the distance measure. Therefore, the KNN algorithm is suitable for applications for which sufficient domain knowledge is available. This knowledge supports the selection of an appropriate measure.
The KNN algorithm is a type of lazy learning, where the computation for the generation of the predictions is deferred until classification. Although this method increases the costs of computation compared to other algorithms, KNN is still the better choice for applications where predictions are not requested frequently but where accuracy is important.
KNN performs best with a low number of features. When the number of features increases, then it requires more data. When there's more data, it creates an overfitting problem because no one knows which piece of noise will contribute to the model. KNN performs better with low dimensionality (as demonstrated by a study by Gu and Shao in 2014).

References:

https://neptune.ai/blog/knn-algorithm-explanation-opportunities-limitations
https://www.ibm.com/docs/en/ias?topic=knn-usage
https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine learning
https://www.kaggle.com/code/hasansezertaan/machine-learning-dersleri-s-n-fland-rma
https://pyimagesearch.com/2016/08/08/k-nn-classifier-for-image-classification/