CSC 635 Data Mining

Assignment 3 Report

Submitted to:

Dr. Jamil Saquer

Author:

Sharmin Sultana

Introduction

In this assignment, we implemented K-Nearest Neighbors (KNN) algorithm in two MNIST datasets [1]. Among them, one dataset is used for training purpose, and another is utilized for testing the model. Each dataset is about gray-scale images of hand-drawn digits, ranging from zero through nine.

The training dataset (MINST_train.csv) contains 949 data where each data is comprised of 785 columns. The first column, named 'label', indicates the digits from zero to nine. The remaining columns represent the pixel value of associated image and values are between 0 to 255, inclusive. The higher the pixel value is, the darker that pixel is. Each pixel column has a labelling like 'pixelx' where x is an integer between 0 and 783, inclusive.

The test dataset (MNIST_test.csv) has the same type of data as the training dataset. A total of 50 data have been stored in the test dataset. The attributes (features and class labels) are like the training dataset.

In this given assignment, our prime intension is to choose the value of K and after that we need to build the KNN model that will classify the test objects based on its learning from the training dataset. The percentage of accurate prediction and total number of misclassified samples out of total test samples have also been examined in this assignment.

Background

For this task, we used one of the popular distance-based classification algorithms which is K-Nearest Neighbors (KNN) algorithm. Here, K indicates the number of neighbors selected for an unseen object based on their Euclidean Distance.

The Euclidean Distance is determined by using the following formula:

$$\textbf{\textit{Equation}}\ (\mathbf{1}):\ d\ (p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

Here, p and q refer two points in the Euclidean n-space [2].

After measuring the distance of the unseen object with respect to all training objects, KNN selects K shortest distanced neighbors. Then it considers the class labels of that K-neighbors and based on the majority voting, assign new class to that new object.

In choosing the value of K, we should consider few restrictions to avoid poor accuracy. If the value of K is too small, then KNN will include many noisy points during model construction. Again, if K value is larger, number of misclassified samples will be higher i.e., built model may include points from other classes. So, we should choose the value of K carefully by trying different range of values and comparing them with the accuracy rate.

Implementation

To carry on the task, I followed following steps.

1. At the beginning, I imported the libraries that are essential for this task. I used *pandas* for loading the datasets and *numpy* for mathematical calculations. Since we are not allowed to use any built-in library to implement the KNN model, I just used these two libraries to fulfill my assignment goal.

2. After importing the libraries, I read the datasets (MNIST_train.csv and MNIST_test.csv) by using read_csv() method. Then I separated the features' part and class part from these two datasets.
3. In this phase, I wrote down a function EUdistance that takes two points (pA and pB) as arguments. By using the formula (1), Euclidean Distance of these two points are calculated.
4. A function named KNN is defined in this stage. The feature attribute values of training set (X), class attribute values of training set(y), feature values of test dataset (x_query) and the number of neighbors (k) are the parameters of this function.

```
def kNN(X, y, x_query, k):
```

5. In this function, KNN model is constructed as per the algorithm. To build the model, I first calculated the Euclidean Distances where pA are the values of test data and pB are the values of training data. Each calculated distances are then appended in an array. After that, they are sorted in a ascendent fashion.
6. From the sorted distances, I considered K distances that was done by using array slicing operation.
7. After that, labels of each neighbor were taken into consideration and counted the unique labels and assigned the most appear label as the prediction for the new data that is X_query. This prediction was then returned from the function KNN.
8. After defining the KNN, in this part I call the function. For this I asked the user to give a value of K.

```
k = int(input("K = "))
predictions = []
for i in range(50):
    temp = kNN(X, y, X_test[i], k)
    predictions.append(temp)
print ('.....................................................')
predictions = np.array(predictions)
for pred in range (len(predictions)):
    print('Desired class: %d computed class: %d' % (y_test[pred], predictions[pr
```

K =

When user gives the value of K i.e., total number of neighbors, it will calculate the desired and computed class of each value of X_query which is basically the feature sets of testing dataset.

9. At this point, accuracy of our model is calculated. For this, I run a checking process where it checks the predicted label with the desired label. If both labels match, then do sum and divide the total matched labels with the total length of the predicted dataset. After running this checking process, for K=4, my constructed model gives 86% accurate data.
10. Number of misclassified samples were calculated with a simple formula.

$$Number\ of\ Missclassified\ Sample = Total\ Number\ of\ Samples - (Accuracy\ Rate * Total\ Number\ of\ Samples)$$

11. As the final requirement of the assignment, I calculated the total number of test samples by simply calling the built-in function len(X_test) that tells us that there are 50 samples in total.

Experimental Result

I choose the 4 as the number of neighbors, and after giving 4 as input, it will compute the desired and computed classes for all the test samples by calling the function KNN. Sample screenshot is as followed:

```
K = 4
.............................................
Desired class: 0 computed class: 0
Desired class: 0 computed class: 0
Desired class: 0 computed class: 0
Desired class: 0 computed class: 0
Desired class: 0 computed class: 0
Desired class: 1 computed class: 1
Desired class: 1 computed class: 1
Desired class: 1 computed class: 1
Desired class: 1 computed class: 1
Desired class: 1 computed class: 1
Desired class: 2 computed class: 8
Desired class: 2 computed class: 2
Desired class: 2 computed class: 2
Desired class: 2 computed class: 6
Desired class: 2 computed class: 2
Desired class: 3 computed class: 3
Desired class: 3 computed class: 3
Desired class: 3 computed class: 3
Desired class: 3 computed class: 3
Desired class: 3 computed class: 3
Desired class: 4 computed class: 2
Desired class: 4 computed class: 4
Desired class: 4 computed class: 4

show more (open the raw output data in a text editor) ...

Desired class: 9 computed class: 9
Desired class: 9 computed class: 7
Desired class: 9 computed class: 9
Desired class: 9 computed class: 9
Desired class: 9 computed class: 9
```

Figure 1: Value of K and desired-computed class result

After that I calculated the accuracy score and the total number of total misclassified samples and the total number of test samples.

```
Accuracy Rate: 86.0 %
Number of misclassified test samples: 7
Total number of test samples: 50
```

Figure 2: Accuracy and sample results

Conclusion

In this assignment, we are assigned to implement KNN algorithm from scratch by using python. In the KNN algorithm, there is two methods of choosing label for the unknown data. One is majority voting, and another is weighted voting. In this task, simple voting which is majority voting is used. The implementation will give us precise idea about how KNN works step by step.

References

[1] "Digit Recognizer." *Kaggle.com*, www.kaggle.com/c/digit-recognizer/data

[2] "Euclidean Distance Formula - Derivation, Examples." *Cuemath*, www.cuemath.com/euclidean-distance-formula/