CSC <535/635> Data Mining

Assignment <2> Report Collection

Submitted to:

Dr. Jamil Saquer

Author(s):

*<Junya Zhao>*

**Report**

**Introduction**

This experiment used the MNIST ("Modified National Institute of Standards and Technology") dataset, which release in 1999. This classic dataset of handwritten images has served as the basis for benchmarking classification algorithms. The data files train.csv and test.csv contain gray-scale images of hand-drawn digits, from zero through nine. Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255, inclusive. The training data set has 785 columns. The first column, called "label", is the digit that was drawn by the user. The rest of the columns contain the pixel-values of the associated image.

**Background**

K-nearest neighbor (k-NN) classification is a non-parametric classifier, which is based on measuring the distances between the test data and each of the training data to decide the final classification output. This Algorithm is based on feature similarity: How closely out-of-sample features resemble our training set determines how we classify a given data point. To measure the distance between two samples *X* and *Z*, this experiment used the Euclidean distance which is the most widely used one. Let X and *Z* are represented by feature vectors *X* = (*x*1, *x*2, …, *xm*) and *Z* = (*y*1, *y*2, …, *ym*), where *m* is the dimensionality of the feature space. To calculate the distance between *X* and *Z*, the normalized Euclidean metric is generally used by. In addition, choosing *k* can be considered as one of the most important factors of this model that can strongly influence the quality of predictions, so this experiment used the k value ranging from 1 to 20 and try to find better result.

**Implementation**

This experiment started with data processing, using the [csv](https://docs.python.org/2/library/csv.html#module-csv) module and its functions to read the dataset and get the list of data values without the header(first row). Importantly, because the original dataset were ordered by labels, which means each sample always near to other samples which are under same class, so all the data were shuffled before feeding into algorithm. Nest, using MNIST\_train.csv, MNiST\_test.csv and value k as input, the algorithm calculated the distance between samples, sorted the distance, implemented weighed voting, and calculated the accuracy of prediction and number of misclassified samples. The class of each of the K neighbors is multiplied by a weight to the inverse of the distance from that point to the given test point. This ensures that nearer neighbors contribute more to the final vote than the more distant ones. Moreover, without knowing how to select the k value, this experiment set up a for loop of k ranged for 1 to 20, the result will print all the result sequentially from k=1 to k =20.

**Experimental Setup and Results**

The result show that when k= 4，it achieved the best performance, this are 7 out of 50 samples were miss-classified, the accuracy is 86%. When k=18, there 25 samples are misclassified, the accuracy is only 50%. According the result, we can see the k value have really important influence on the result, and with increasing of k-value, the result become worse . Obviously, the more neighbors for one sample are included, the less accuracy will get.

**Conclusion**

In conclusion, the experiment achieved best result that accuracy is 86% when k=4. In order to make an improvement for this algorithm, changing the distance metric for different data may help improve the accuracy of the algorithm. Also, rescaling the data makes the distance metric more meaningful.

**Improvement (extra thinking)**

For some groups of dataset that have high capacities are at the center of areas dense in prototypes of the same class, so it will be good to remove some of the prototypes with high attractive capacity from the groups. However, it is hard to know which group is over a certain threshold, because capacity of a prototype is determined not by itself but mainly by others, so the attractive capacities change dynamically. When certain patterns are eliminated from the template, the attractive capacities of the remaining patterns may either increase or decrease depending on the distribution of the patterns in feature space.

**Code**

*##########################################################################*

*# Program: Hw2\_K Nearest Neighbors (KNN)*

*# Programmed By: Junya Zhao*

*# Input:*

*# trainData: training dataset*

*# k:number of neighbors*

*# testData :new input tuple to classfiy*

*#*

*# Output: class to which t is assigned*

*#*

*# display : value of K;*

*# both the desired class and the computed class for each sample*

*# desired class is the class label as given in the data set*

*# computed class is what produces as the output for the sample*

*# the accuracy rate*

*# number of misclassified test samples,*

*# total number of test samples*

*#Trace folder: \\trace\Class\CSC-535-635\001\Junya009\hw2*

*############################################################################*

*#---------------------------------Imports--------------------------------------*

*import csv*

*import math*

*from random import shuffle*

*from operator import itemgetter*

*#---------------------------------Functions------------------------------------*

*#get the dataset*

*def GetDataset(filename, isTestData = True):*

*with open(filename, 'r') as file:*

*csv\_reader = csv.reader(file)*

*data = list(csv\_reader)# get a list of all values*

*del(data[0]) # delete the header*

*data = [list(map(int, d)) for d in data]*

*if not isTestData:*

*shuffle(data) #shuffle the data, in order to get the different order of samples*

*#to make sure the neighbors does not always have same label*

*return data*

*#Calculates the distance between samples*

*def Distance(testData, trainingData):*

*distance= 0*

*for i in range(len(testData)):*

*distance += math.pow(testData[i] - trainingData[i],2)*

*return math.sqrt(distance)*

*#get the accuracy of the result*

*def accuracy(TestData, outputset):*

*rightLabel= 0*

*WrongLabel = 0*

*for i in range(len(TestData)):*

*if TestData[i][0] == outputset[i]:*

*rightLabel +=1*

*else:*

*WrongLabel +=1*

*return [(rightLabel/len(outputset)\*100), WrongLabel]*

*#get the weights for closest neighbors*

*def CalVotes(data):*

*for i in data:*

*vote = 1/i[1]*

*i.append(vote)*

*data.sort(key=itemgetter(2))*

*return data[0][0]*

*# KNN Algorithm'''*

*def KNNclassify(testData, trainingData, k) :*

*Outputdata= list()*

*for newIN in testData:*

*distance = list()*

*for train in trainingData:*

*distance.append([train[0], Distance(newIN[1:], train[1:])])*

*distance.sort(key=itemgetter(1))*

*neighbors = [distance[i] for i in range(k)]*

*result = CalVotes(neighbors)*

*Outputdata.append(result)*

*print('Desired Class = ', newIN[0], "Computed Class = ", result)*

*accuracy\_result = accuracy(testData, Outputdata)*

*print ('Accuracy Rate = ', accuracy\_result[0])*

*print('Number of mis-classified test samples = ', accuracy\_result[1])*

*print('Total number of test samples = ', len(testData))*

*#------------------------------------------------------------------------------*

*#---------------------------------Program Main---------------------------------*

*if \_\_name\_\_ == '\_\_main\_\_':*

*Traing = 'MNIST\_train.csv'*

*TestDataSet = 'MNIST\_test.csv'*

*trainingData = GetDataset(Traing, False)*

*testingData = GetDataset(TestDataSet)*

*for k in range(1,20):*

*print('\n\nK = ', k)*

*KNNclassify(testingData,trainingData, k)*

*#---------------------------------End of Program-------------------------------*