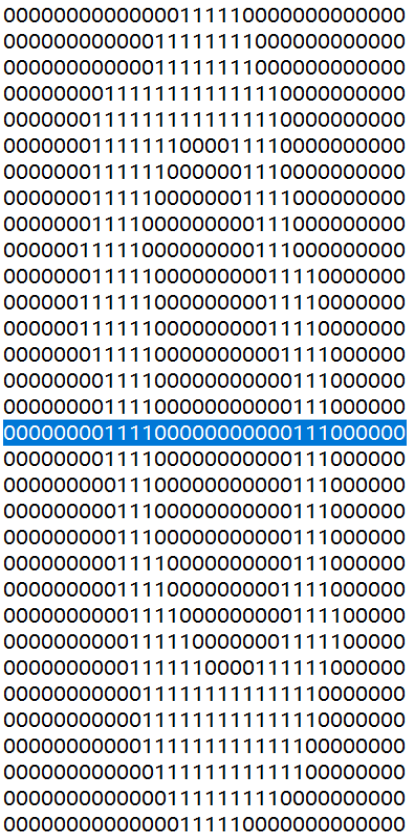
Assignment3 CSC1002 Design

1. Program structure and flow:
   a) The main process:

      Training part: (for digit-training.txt)

      First, load the data. Parse all digits from training file and store all digits (as vectors) in dictionary g_dataset. When create the digit vector, in my solution, it just read 8 horizontal lines and 8 vertical lines for each number, for every drawn line, count the weight of the stroke and points of intersection. For example, as the following picture shows:



      For this horizontal line, it counts the number of “1” as the weight of the stroke(Here is 7) and here the points of intersection is 2. Let’s call them “number characteristics”

      So for every lines it will get two “number characteristics”, for every number it pick 8 horizontal lines and 8 vertical lines, finally it will get (8+8)*2 = 32 “number characteristics” for every number. It stores these 32 “number characteristics” as a vector.

      Testing part (for digit-testing.txt)

      Use KNN method to compute the distance for every data in dictionary g_datase to predict the number, and check the accuracy of the prediction.

      Final predict part(for digit-predict.txt)

      Use KNN method to compute the distance for every data in dictionary g_datase to predict the number and print it out.

   b) KNN model:

      K value = 7

How the closest neighbors are determined:

Distance += abs(vi-wi)

Then choose the 7 smallest-distance numbers as the closest neighbors

The rule(s) used in making the prediction:

In the 7 smallest-distance numbers, the number that occurs most is the closest number.

c) One strategy in reducing the kNN computation time:

In my solution, it takes 12 seconds in validation part to check all 943 numbers. The accuracy reach 95.97%

If we select less lines for every number, for example, 12 lines(Here I select 16 lines originally), there will be less "number characteristics" in each vector, and the computation speed will be faster(9seconds), but the accuracy will reduce a little bit.(93.74%)

2. Python objects (global variables)

g_dataset = {0:[[1,0,⋯.],⋯,[0,1⋯]],1:[[⋯]]⋯⋯}

It stores all the testing data.

g_test_good = {}

It stores the number of times that it makes a right prediction for each digit number.

g_test_bad = {}

It stores the number of times that it makes a wrong prediction for each digit number.

```
NUM_ROWS = 32                       # number of rows
NUM_COLS = 32                       # number of columns
DATA_TRAINING = 'digit-training.txt'  # training data
DATA_TESTING = 'digit-testing.txt'    # testing data
DATA_PREDICT = 'digit-predict.txt'    # predict data
KNN_NEIGHBOR = 7                    # kNN parameter
val                                 # value of the number
new_vec                             # vector that has 32 "number characteristics"
dist                                # the distance of the vectors
```

3. Functions

Load data part:

read_digit(p_fp): Convert next digit from input file as a vector. Return (digit, vector) or (-1, '') on end of file. Finally it will return val, new_vec.

load_data(p_filename=DATA_TRAINING): Parse all digits from training file and store all digits (as vectors) in dictionary g_dataset.

KNN models parts:

knn(p_v, size=KNN_NEIGHBOR): Given a digit vector, returns the k nearest neighbor by vector distance.

knn_by_most_common(p_v): Based on the knn Model (nearest neighhor), return the target value.

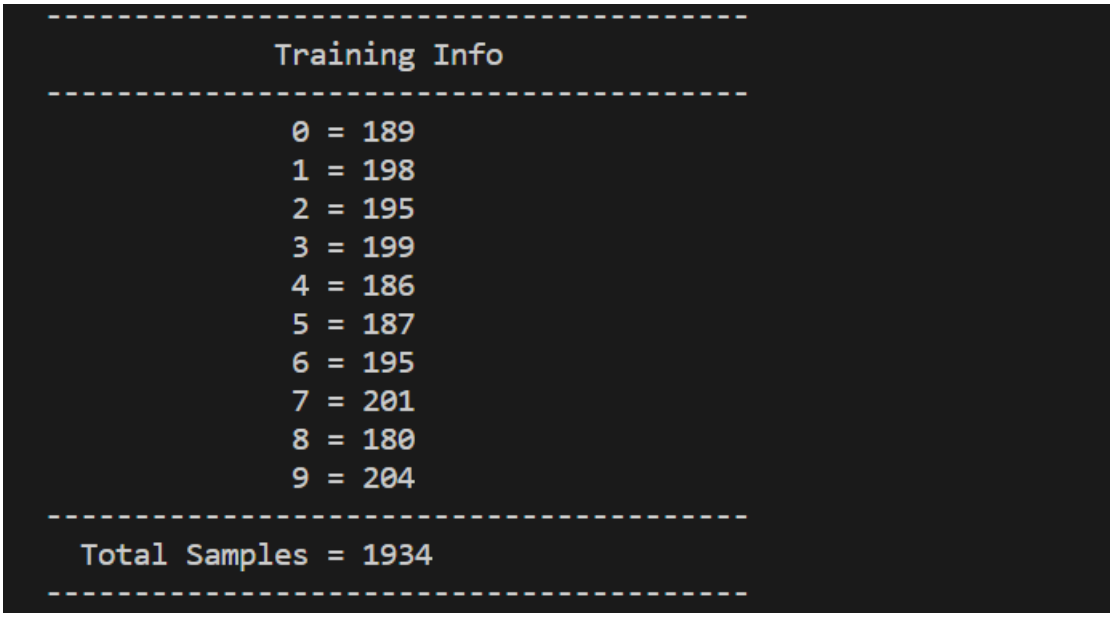
Predict part:
predict(p_filename): Make prediction based on kNN model.Parse each digit from the predict file.
final_predict(p_filename=DATA_PREDICT): Make prediction based on kNN model.Parse each digit from the predict file and print the predicted value.

Validate part:
validate(p_filename=DATA_TESTING): Compile an accuracy report by comparing the data set with every digit from the testing file.

Vector part:
distance(v, w): Return distance between vectors v & w

Report part:
show_info(): Show info for training data set
show_test(start="????", stop="????"): Show test results

4. Output



```
----------------------------------------
Beginning of Validation @  2019-04-25 23:25:00.436499
----------------------------------------
              Testing Info
----------------------------------------
              0 = 100,   0,  100%
              1 = 94 ,   0,  100%
              2 = 88 ,   5,   95%
              3 = 102,   3,   97%
              4 = 86 ,   1,   99%
              5 = 76 ,   5,   94%
              6 = 95 ,   0,  100%
              7 = 87 ,   3,   97%
              8 = 97 ,  12,   89%
              9 = 80 ,   9,   90%
----------------------------------------
       Accuarcy = 95.97%
  Corrent/Total = 905/943
----------------------------------------
End of Validation @  2019-04-25 23:25:12.774991
----------------------------------------
```

```
              7 = 87 ,   3,   97%
              8 = 97 ,  12,   89%
              9 = 80 ,   9,   90%
----------------------------------------
       Accuarcy = 95.97%
  Corrent/Total = 905/943
----------------------------------------
End of Validation @  2019-04-25 23:25:12.774991
----------------------------------------
Final predict:
7
5
2
1
8
2
9
9
5
4
3
6
```