# HomeWork1 (Solutions)

**Note: That solutions are in the reverse order of the problems order in the HW1 assignment**

**Question 3:** (KNN for handwriting digit recognition)

***Solution:***

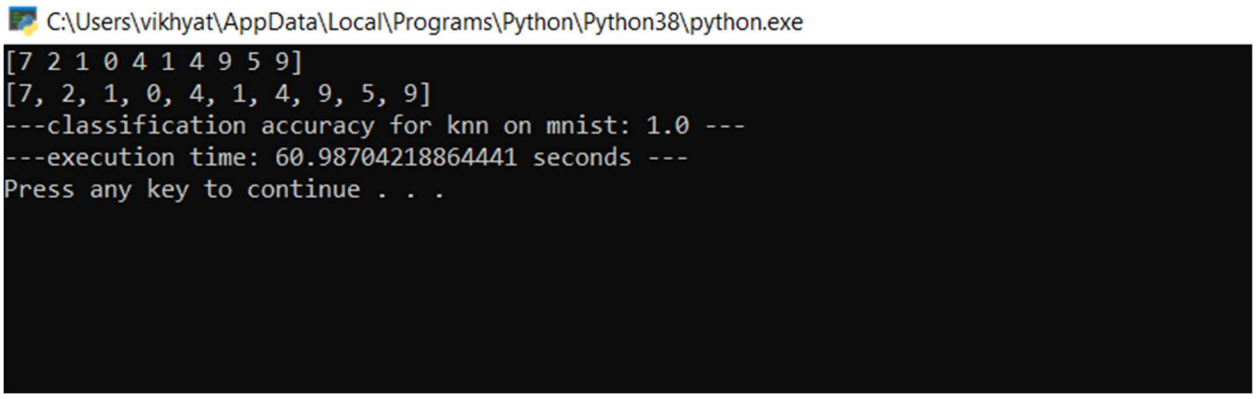
**Python Code with Comments**

```
import math
import numpy as np
from download_mnist import load
import operator
import time
# classify using kNN
#x_train = np.load('../x_train.npy')
#y_train = np.load('../y_train.npy')
#x_test = np.load('../x_test.npy')
#y_test = np.load('../y_test.npy')
x_train, y_train, x_test, y_test = load()
x_train = x_train.reshape(60000,28,28)
x_test  = x_test.reshape(10000,28,28)
x_train = x_train.astype(float)
x_test = x_test.astype(float)

#Function for Knn classification

def kNNClassify(newInput, dataSet, labels, k):
        # variable for counting the length of input array of image matrices
        x=0
        # variable for storing the resulting labels for all the testing images
        result_labels=[]
        #upper loop for the number of testing images that is the length of the input array for testing
        while(x < len(newInput)):
            #empty array for storing the distances of the one testing image with 60k training images
            np_distance=np.array([])
            #loop for calculating the distances of the image being tested with the training images
            for y in range(60000):
                sum1=0
                sub=np.subtract(newInput[x],dataSet[y])#first subtraction
                square=np.power(sub,2)#then squaring
                sum1=np.sum(square)#then sum of all the values of the matrices
                #np_distance is having tuples with two entries 1 distance from training image and the label
                np_distance=np.append(np_distance,np.array([math.sqrt(sum1)]), axis=0)
                np_distance=np.append(np_distance,np.array([labels[y]]),axis=0)
            #reshaping into two Dimension array
            np_distance= np_distance.reshape(60000,2)
            np_distance = np_distance[np_distance[:,0].argsort(kind='mergesort')]
            #Then sorting with respect to distance so that lowest distances come up
            j=0
```

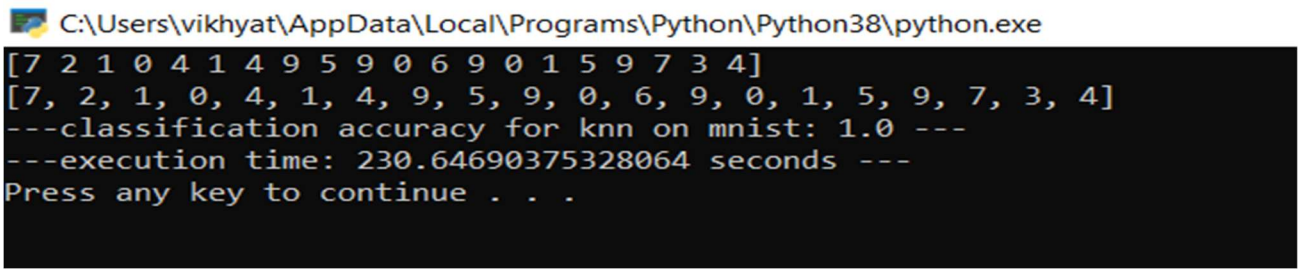
```
            #labels counting as labels are from 0 to 9 for digits
            labels_count=[0,0,0,0,0,0,0,0,0,0]
            #converting the second column of the two d array and then converting into int
            label=np_distance[:,1]
            label=label.astype('int64')


            #For k lowest distance values we count the respective number of labels
            while(j < k):
                labels_count[label[j]]+=1
                j+=1
            #Then finding the maximum value of the label count
            j=0
            max=labels_count[0]
            max_index=0
            while(j < 10):
                if(labels_count[j] > max ):
                    max_index=j
                    max=labels_count[j]
                j+=1
            #then that label will be asigned in the result vector matrices for the corresponding testing image
            result_labels.append(max_index)
            x+=1
            #incrementing value of x
        return result_labels

start_time = time.time()
outputlabels=kNNClassify(x_test[0:10],x_train,y_train,10)
#printing the true labels of the testing images
print(y_test[0:10])
#printing the result after classification
print(outputlabels)
#resulting accuracy
result = y_test[0:10] - outputlabels
result = (1 - np.count_nonzero(result)/len(result))
print ("---classification accuracy for knn on mnist: %s ---" %result)
print ("---execution time: %s seconds ---" % (time.time() - start_time))
```

**Output of this code :**

a. For 10 images



b. For 20 images




**Question 2: (KNN for simple data).**

***Solution:***

**Python Code with Comments**

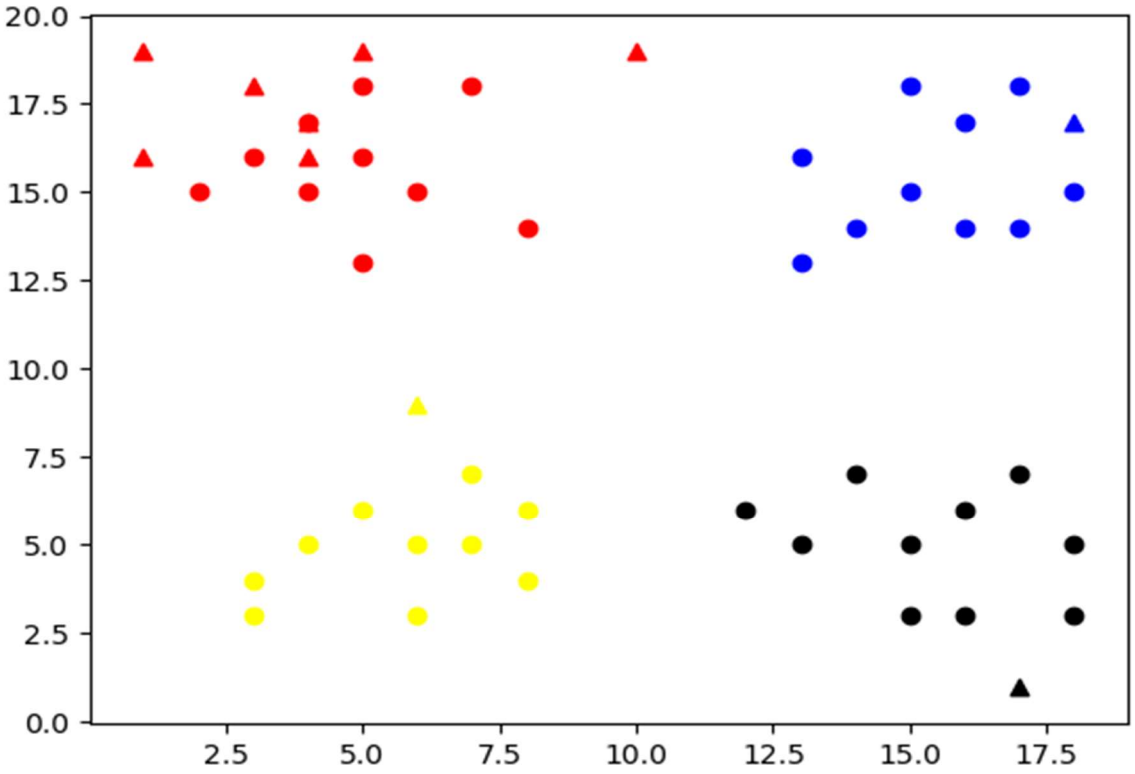
```
import numpy as np
import matplotlib as mpl
mpl.use('Agg')
import matplotlib.pyplot as plt
import math


# load mini training data and labels
mini_train = np.load('knn_minitrain.npy')
mini_train_label = np.load('knn_minitrain_label.npy')

# randomly generate test data
mini_test = np.random.randint(20, size=20)
mini_test = mini_test.reshape(10,2)


# Define knn classifier
def kNNClassify(newInput, dataSet, labels, k):
        #array for storing the resulting labels of the testing data that is the random points being generated
        result=[]
    ########################
    # Input your code here #
    ########################
        x=0
        #x is a counter variable for the loop for the number of testing data which is 10 in our program
        while (x < 10 ):

            np_distance=np.array([])
            #np_distance is an empty np array for distances and their corresponding labels
            for y in range(40):
                sum1=0
                sub=np.subtract(newInput[x],dataSet[y])#first subtract the testing point from the training point
                square=np.power(sub,2)#squaring the diffrence
                sum1=np.sum(square)# here summing to calculate x^2 + y^2
```

```
            np_distance=np.append(np_distance,np.array([math.sqrt(sum1)]), axis=0)# storing the distance
            np_distance=np.append(np_distance,np.array([labels[y]]),axis=0)#storing the label
        #reshaping the array to 40 into 2 two d array
        np_distance= np_distance.reshape(40,2)
        np_distance = np_distance[np_distance[:,0].argsort(kind='mergesort')]
        #sorting distances

        j=0
        #As there are four labels assigned
        labels_count=[0,0,0,0]
        #coverting the column of numpy array into the int as float is their default datatype
        label=np_distance[:,1]
        label=label.astype('int64')



        #counting the labels of the k smallest or nearest distance training points
        while(j < k):
            labels_count[label[j]]+=1
            j+=1

        j=0
        max=labels_count[0]
        #finding the maximum out of them
        max_index=0
        while(j < 4):
            if(labels_count[j] > max ):
                max_index=j
                max=labels_count[j]
            j+=1
        #storing the classified label in the result
        result.append(max_index)
        x+=1

    ####################
    # End of your code #
    ####################
        return result

outputlabels=kNNClassify(mini_test,mini_train,mini_train_label,3)

print ('random test points are:', mini_test)
print ('knn classfied labels for test:', outputlabels)

# plot train data and classfied test data
train_x = mini_train[:,0]
train_y = mini_train[:,1]
fig = plt.figure()
plt.scatter(train_x[np.where(mini_train_label==0)], train_y[np.where(mini_train_label==0)], color='red')
plt.scatter(train_x[np.where(mini_train_label==1)], train_y[np.where(mini_train_label==1)], color='blue')
plt.scatter(train_x[np.where(mini_train_label==2)], train_y[np.where(mini_train_label==2)], color='yellow')
plt.scatter(train_x[np.where(mini_train_label==3)], train_y[np.where(mini_train_label==3)], color='black')

test_x = mini_test[:,0]
test_y = mini_test[:,1]
outputlabels = np.array(outputlabels)
plt.scatter(test_x[np.where(outputlabels==0)], test_y[np.where(outputlabels==0)], marker='^', color='red')
plt.scatter(test_x[np.where(outputlabels==1)], test_y[np.where(outputlabels==1)], marker='^', color='blue')
```

```
plt.scatter(test_x[np.where(outputlabels==2)], test_y[np.where(outputlabels==2)], marker='^', color='yellow')
plt.scatter(test_x[np.where(outputlabels==3)], test_y[np.where(outputlabels==3)], marker='^', color='black')

#save diagram as png file
plt.savefig("miniknn.png")
```

**Output of this code :**

```
C:\Users\vikhyat\data_structures_algo\deep_learning_homework>python miniknn.py
random test points are: [[ 1 19]
 [ 4 17]
 [ 4 16]
 [ 1 16]
 [17  1]
 [18 17]
 [10 19]
 [ 5 19]
 [ 3 18]
 [ 6  9]]
knn classfied labels for test: [0, 0, 0, 0, 3, 1, 0, 0, 0, 2]
```



Ques1→

Ans1→ **Numerical Computation using KNN Algorithm**

KNN stands for K- Nearest Neighbour algorithm used for Machine Learning.

Training Dataset :→

Class A: ① (0,1,0) ② (0,1,1) ③ (1,2,1) ④ (1,2,0)

Class B: ⑤ (1,2,2) ⑥ (2,2,2) ⑦ (1,2,-1) ⑧ (2,2,3)

Class C: ⑨ (-1,-1,-1) ⑩ (0,-1,-2) ⑪ (0,-1,1) ⑫ (-1,-2,1)

A, B, C are the Labels for this Supervised type of Machine Learning

Objective:- To Find the Classified Label for Test Data i.e. vector (1,0,1) when K = 1,2,3 respectively.

(Step 1)→ To calculate the Distance from all the Training Data [ie. Note that Training Data Points are marked from ① to ⑫ as shown above] of our Test Data as mentioned above

L2 Distance from ① = (0,1,0) and (1,0,1)

$$= \sqrt{(1-0)^2 + (0-1)^2 + (1-0)^2}$$

$$= \sqrt{3}$$

② $(0,1,1)$ and $(1,0,1)$

$= \sqrt{(1-0)^2 + (0-1)^2 + (1-1)^2} = \sqrt{2}$

③ $(1,2,1)$ and $(1,0,1)$

$= \sqrt{(1-1)^2 + (0-2)^2 + (1-1)^2} = 2 = \sqrt{4}$

④ $(1,2,0)$ and $(1,0,1)$

$= \sqrt{(1-1)^2 + (0-2)^2 + (1-0)^2} = \sqrt{5}$

⑤ $(1,2,2)$ and $(1,0,1)$

$= \sqrt{(1-1)^2 + (0-2)^2 + (1-2)^2} = \sqrt{5}$

⑥ $(2,2,2)$ and $(1,0,1)$

$= \sqrt{(1-2)^2 + (0-2)^2 + (1-2)^2} = \sqrt{6}$

⑦ $(1,2,-1)$ and $(1,0,1)$

$= \sqrt{(1-1)^2 + (0-2)^2 + (1+1)^2} = \sqrt{8}$

⑧ $(2,2,3)$ and $(1,0,1)$

$= \sqrt{(1-2)^2 + (0-2)^2 + (1-3)^2}$

$= \sqrt{9} = 3$

⑨ $(-1,-1,-1)$ and $(1,0,1) = \sqrt{(2)^2 + 1^2 + 2^2}$

$= \sqrt{9} = 3$

(10) $(0,-1,-2)$ and $(1,0,1)$

$= \sqrt{1^2+1^2+3^2} = \sqrt{11}$

(11) $(0,-1,1)$ and $(1,0,1)$

$\sqrt{1^2+1^2+0^2} = \sqrt{2}$

(12) $(-1,-2,1)$ and $(1,0,1)$

$= \sqrt{2^2+2^2+0^2} = \sqrt{8}$

Step2:→ Sort and find the K-nearest Neighbours and then based on highest votes among K define the Class for Test Data

For various values of K

K=1 :- The most nearest neighbour

Least Distance is :- $\sqrt{2}$

From:- Point (2) $(0,1,1)$ :- Class B

Point (11) $(0,-1,1)$ :- Class C

So, Class A or Class C can be the answer But as — we consider the order — Class B can be assigned.

K=2 = Two most nearest points are or neighbours

(a) Class A :- (2) (0,1,1) with Dist :- $\sqrt{2}$

(b) Class C :- (11) (0,-1,1) with Dist :- $\sqrt{2}$

Here - as explained earlier

Two nearest neighbours are as above.

so Class A or Class C can be choosen has equal votes (ie 1 & 1)

So Class A can be choosen if choosing from orders of Labels.

K=3 = Three most nearest points

(a) Class A:- (2) (0,1,1) with Dist:- $\sqrt{2}$

(b) Class C:- (11) (0,-1,1) with Dist:- $\sqrt{2}$

(c) Class A:- (1) (0,1,0) with Dist:- $\sqrt{3}$

Here, we can see that when we take 3 Nearest Neighbours then —

Votes for Class A are more (2) out of (3)

Hence - Test Data is Classified as Label A ie. Class A

Conclusion:—

The above problem shows:—

(1) Computation of KNN algorithm.

(2) How proper selection of K is required for proper Classification.