Homework 1 Introduction to Deep Learning.

Swapnil Kamate

Sk2181

Solution 1:

Introduction to deep learning HW 1.

Swapnil Kamate

SK2181

Class A : $\overset{D_1}{(0,1,0)}, \overset{D_2}{(0,1,1)}, \overset{D_3}{(1,2,1)}, \overset{D_4}{(1,2,0)}$

Class B : $(1,2,2)^{D_5}; (2,2,2)^{D_6}; (1,2,-1)^{D_7}; (2,2,3)^{D_8}$

Class C : $(-1,-1,-1)^{D_9}; (0,-1,-2)^{D_{10}}; (0,-1,1)^{D_{11}}; (-1,-2,1)^{D_{12}}$

Test Data = $(1,0,1)$

$$L_2 \text{ distance} = \sqrt{(x_2-x_1)^2+(y_2-y_1)^2+(z_2-z_1)^2}$$

So

$$A \begin{cases} D_1 = \sqrt{1+1+1} = \sqrt{3} = 1.73 \rightarrow \\ D_2 = \sqrt{1+1+0} = \sqrt{2} = 1.414 \rightarrow \\ D_3 = \sqrt{0+4+0} = \sqrt{4} = 2 \\ D_4 = \sqrt{0+4+1} = \sqrt{5} = 2.236 \end{cases}$$

$$B \begin{cases} D_5 = \sqrt{0+4+1} = \sqrt{5} = 2.236 \\ D_6 = \sqrt{1+4+1} = \sqrt{6} = 2.449 \\ D_7 = \sqrt{0+4+4} = \sqrt{8} = 2.828 \\ D_8 = \sqrt{1+4+4} = \sqrt{9} = 3 \end{cases}$$

$$C \begin{cases} D_9 = \sqrt{4+1+4} = \sqrt{9} = 3 \\ D_{10} = \sqrt{1+1+9} = \sqrt{11} = 3.3166 \\ D_{11} = \sqrt{1+1+0} = \sqrt{2} = 1.414 \rightarrow \\ D_{12} = \sqrt{4+4+0} = \sqrt{8} = 2.8284 \end{cases}$$

Result:

When K=1 We need to consider only one lowest distance. So, we could see the lowest distance(L2) is 1.414 which is both d2 and d11. So, when value of K =1 then our test data can fall into either Class A or Class C.

When K = 2 Again, in this case we need to consider two lowest distances which are again surprisingly 1.414 and 1.414 which are d2 and d11. So, when K = 2 our test data can fall into either Class A or Class C.

When K = 3, we need to consider 3 lowest distances and according to our results the three lowest distances are d2 = 1.414 -> Class A, d11 = 1.414 -> Class C and d1 = 1.73 -> Class A. We could see two of our distances fall in the Class A so, when K =3 I would say our test data can be classified as Class A result.

Solution 2:

Source code:

```
import numpy as np

import matplotlib as mpl

mpl.use('Agg')

import matplotlib.pyplot as plt


# 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):

    result=[]


    # compute L2 distance for all test and train samples
```

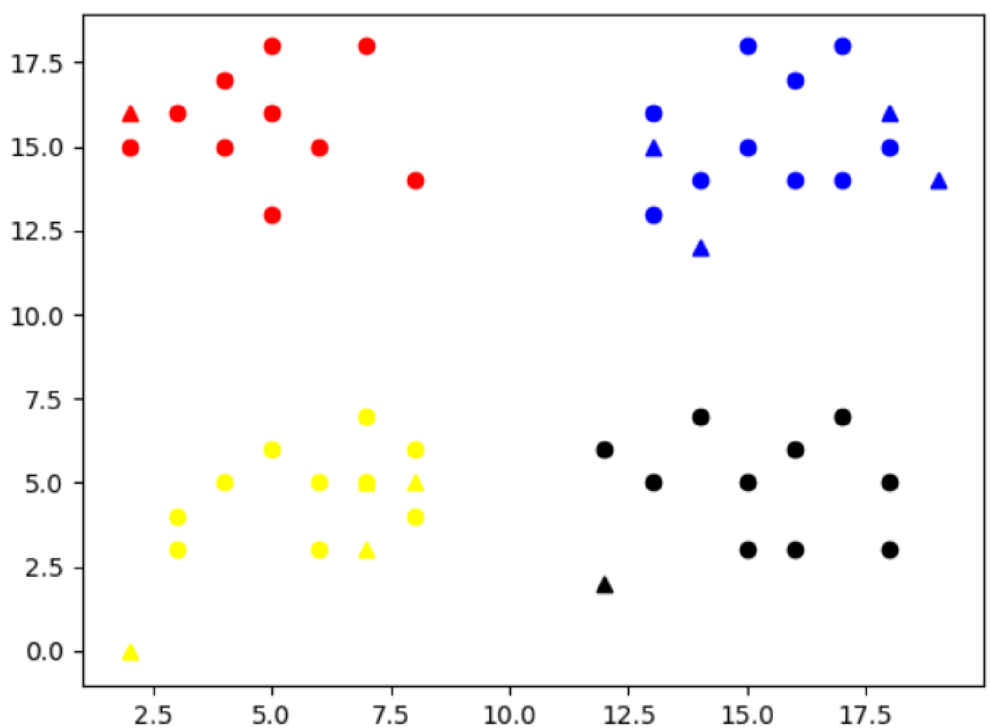
```
    distances = []
    for item1 in newInput:
        d = []
        for item2 in dataSet:
            distance = np.sqrt(np.sum((item1-item2)**2)) #calclulating L2 distance
            d.append(distance)
        distances.append(d)
    print(distances)
    # decide which class the test samples belong in

    for i in range(len(newInput)):
        label_value = np.zeros(4)  # creating an array for lable values
        knn_indices = np.argsort(distances[i])[:k]
        for j in range(len(knn_indices)):
            label = labels[knn_indices[j]] #incrementing the count after getting each 'K'
            label_value[label]+=1
        result.append(np.argmax(label_value))

    return result

outputlabels=kNNClassify(mini_test,mini_train,mini_train_label,4)

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")
```

Screenshot of code:

```python
# Define knn classifier
def kNNClassify(newInput, dataSet, labels, k):
    result=[]

    # compute L2 distance for all test and train samples
    distances = []
    for item1 in newInput:
        d = []
        for item2 in dataSet:
            distance = np.sqrt(np.sum((item1-item2)**2)) #calclulating L2 distance
            d.append(distance)
        distances.append(d)
    print(distances)
    # decide which class the test samples belong in

    for i in range(len(newInput)):
        label_value = np.zeros(4)  # creating an array for lable values
        knn_indices = np.argsort(distances[i])[:k]
        for j in range(len(knn_indices)):
            label = labels[knn_indices[j]] #incrementing the count after getting each 'K'
            label_value[label]+=1
        result.append(np.argmax(label_value))

    return result

outputlabels=kNNClassify(mini_test,mini_train,mini_train_label,4)

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

Output of plotted Image



CMD Output:

```
random test points are: [[ 4  6]
 [14  3]
 [15  4]
 [ 3 14]
 [14 12]
 [17 17]
 [18 15]
 [19  9]
 [ 5  3]
 [14  8]]
knn classfied labels for test: [2, 3, 3, 0, 1, 1, 1, 3, 2, 3]

C:\Users\Swapnil\Desktop\HW1>date
The current date is: Mon 02/22/2021
Enter the new date: (mm-dd-yy)
```

Solution 3:

Source Code:

```
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)
def kNNClassify(newInput, dataSet, labels, k):

    result=[]

    # compute L2 distance for all test and train samples
    distances = np.zeros((len(newInput), len(dataSet)))
    for i in range(len(newInput)):
        for j in range(len(dataSet)):
            distance = np.sqrt(np.sum((newInput[i]-dataSet[j])**2))
            distances[i, j] = distance
    # decide which class the test samples belong in

    for i in range(len(newInput)):
        label_value = np.zeros(10)
        knn_indices = np.argsort(distances[i])[:k]
        for j in range(len(knn_indices)):
            label = labels[knn_indices[j]]
            label_value[label]+=1
        result.append(np.argmax(label_value))

    return result

start_time = time.time()
outputlabels=kNNClassify(x_test[0:20],x_train,y_train,10)
result = y_test[0:20] - outputlabels
result = (1 - np.count_nonzero(result)/len(outputlabels))
print ("---classification accuracy for knn on mnist: %s ---" %result)
print ("---execution time: %s seconds ---" % (time.time() - start_time))
```

Screenshot of Code:

C:\Users\Swapnil\Desktop\knn.py • - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

practise.py | first.html | miniknn.py | knn.py | download_mnist.py

```python
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)
def kNNClassify(newInput, dataSet, labels, k):

    result=[]

    # compute L2 distance for all test and train samples
    distances = np.zeros((len(newInput), len(dataSet)))
    for i in range(len(newInput)):
        for j in range(len(dataSet)):
            distance = np.sqrt(np.sum((newInput[i]-dataSet[j])**2))
            distances[i, j] = distance
    # decide which class the test samples belong in

    for i in range(len(newInput)):
        label_value = np.zeros(10)
        knn_indices = np.argsort(distances[i])[:k]
        for j in range(len(knn_indices)):
            label = labels[knn_indices[j]]
            label_value[label]+=1
        result.append(np.argmax(label_value))

    return result

start_time = time.time()
outputlabels=kNNClassify(x_test[0:20],x_train,y_train,10)
result = y_test[0:20] - outputlabels
result = (1 - np.count_nonzero(result)/len(outputlabels))
print ("---classification accuracy for knn on mnist: %s ---" %result)
print ("---execution time: %s seconds ---" % (time.time() - start_time))
```

Output:

```
C:\Users\Swapnil\Desktop\HW1>python knn.py
---classification accuracy for knn on mnist: 1.0 ---
---execution time: 13.047618627548218 seconds ---

C:\Users\Swapnil\Desktop\HW1>
```