# Homework 1 Introduction to Deep Learning.

# **Swapnil Kamate**

# Sk2181

# **Solution 1:**

Introduction to deep leading HD 1.  (Nagan't Karnett  SK2(81)  (Land A: (0,1,0), (0,1,1), (1,2,1), (1,2,0)  (Land B: (1,2,1), (2,2,2), (1,2,1), (1,2,1), (2,		
Second   Kamate   Sk218   Class A: $(0,1,0)$ , $(0,1,1)$ , $(1,2,1)$ , $(1,2,0)$ Class B: $(1,2,2)^{37}$ $(2,2,2)^{37}$ $(1,2,-1)^{37}$ $(2,2,3)^{28}$ Class C: $(-1,-1,-1)^{29}$ $(0,-1,-2)^{29}$ $(0,-1,1)^{29}$ $(-1,-2,1)^{29}$ Feel Date = $(1,0,1)$ Ly distance = $\sqrt{8}$ , $-n$ <sup>2</sup> + $(x-y)^2$ + $(x_2-2)^2$ $\sqrt{8}$ $\sqrt{9}$ = $\sqrt{1+1+0}$ = $\sqrt{2}$ = $\sqrt{1+1+0}$ = $\sqrt{2}$ = $\sqrt{1+1+0}$ = $\sqrt{2}$ = $\sqrt{1+1+0}$ = $\sqrt{2}$ = $2$		1
Graphit Kamate  SK2 [8]  Class A: $(0,1,0), (0,1,1), (1,2,1), (1,2,0)$ Class B: $(1,2,2)^{3}, (2,2,2)^{3}, (1,2,-1)^{2}, (2,2,3)^{-2}$ Class C: $(-1,-1,-1)^{2}, (0,-1,-2)^{3}, (0,-1,1)^{3}, (-1,-2,1)^{0}$ Test Date = $(1,0,1)$ Ly distance = $\sqrt{R_2 - n}$ $\sqrt{2} + (x_1 - y_1)^{2} + (x_2 - z_1)^{2}$ So $\left( \frac{1}{2} + \frac{1}{2$	Introduction to dean booking HW 1.	6 5
Class A: $(0,1,0)$ , $(0,1,1)$ , $(1,2,1)$ , $(1,2,0)$ Class B: $(1,2,2)^{37}$ $(2,2,2)^{23}$ $(1,2,-1)^{27}$ $(2,2,3)^{26}$ Class C: $(-1,-1,-1)^{29}$ $(0,-1,-2)^{29}$ $(0,-1,1)^{29}$ $(-1,-2,1)^{012}$ Feat Data = $(1,0,1)$ Ly distance = $\sqrt{(0,-1)^2 + (0,-1)^2 + (0,-1,1)^{29}}$ $(-1,-2,1)^{012}$ $\sqrt{(0,-1,-1)^2 + (0,-1)^2 + (0,-1,1)^{29}}$ $(-1,-2,1)^{012}$ $\sqrt{(0,-1,-1)^2 + (0,-1,1)^2}$ $\sqrt{(0,-1,1)^{29}}$ $(-1,-2,1)^{012}$ $\sqrt{(0,-1,-1)^2 + (0,-1,1)^2}$ $\sqrt{(0,-1,1)^{29}}$ $(-1,-2,1)^{012}$ $\sqrt{(0,-1,-1)^2 + (0,-1,-1)^2}$ $\sqrt{(0,-1,-1)^{29}}$ $(0,-1,-1)^{2$	(wasuit Kamate	
Test Data = $(1,0,1)$ La distance = $\sqrt{(2-n)^2 + (2-2)^2 + (2-2)^2}$ So $\begin{cases} D_1 = \sqrt{1+1+1} & = \sqrt{3} = 1.73 \rightarrow \\ D_2 = \sqrt{1+1+0} = \sqrt{2} = 1.414 \rightarrow \\ D_3 = \sqrt{0+4+0} = \sqrt{4} = 2 \\ D_4 = \sqrt{0+4+1} = \sqrt{5} = 2.236 \end{cases}$ Dy = $\sqrt{0+4+1} = \sqrt{5} = 2.236$ Dy = $\sqrt{0+4+1} = \sqrt{5} = 2.449$ Dy = $\sqrt{0+4+1} = \sqrt{5} = 2.828$ Dy = $\sqrt{1+4+1} = \sqrt{5} = 2.828$ Dy = $\sqrt{1+4+1} = \sqrt{5} = 3$ C Dy = $\sqrt{1+1+1} = \sqrt{11} = 3.3166$ Dy = $\sqrt{1+1+0} = \sqrt{1} = 3.3166$ Dy = $\sqrt{1+1+0} = \sqrt{1} = 1.414$ Dy = $\sqrt{1+1+0} = \sqrt{1} = 1.414$	cra(v)	
Test Data = $(1,0,1)$ La distance = $\sqrt{(2-n)^2 + (2-2)^2 + (2-2)^2}$ So $\begin{cases} D_1 = \sqrt{1+1+1} & = \sqrt{3} = 1.73 \rightarrow \\ D_2 = \sqrt{1+1+0} = \sqrt{2} = 1.414 \rightarrow \\ D_3 = \sqrt{0+4+0} = \sqrt{4} = 2 \\ D_4 = \sqrt{0+4+1} = \sqrt{5} = 2.236 \end{cases}$ Dy = $\sqrt{0+4+1} = \sqrt{5} = 2.236$ Dy = $\sqrt{0+4+1} = \sqrt{5} = 2.449$ Dy = $\sqrt{0+4+1} = \sqrt{5} = 2.828$ Dy = $\sqrt{1+4+1} = \sqrt{5} = 2.828$ Dy = $\sqrt{1+4+1} = \sqrt{5} = 3$ C Dy = $\sqrt{1+1+1} = \sqrt{11} = 3.3166$ Dy = $\sqrt{1+1+0} = \sqrt{1} = 3.3166$ Dy = $\sqrt{1+1+0} = \sqrt{1} = 1.414$ Dy = $\sqrt{1+1+0} = \sqrt{1} = 1.414$	9 92 93 94	-
Test Data = $(1,0,1)$ La distance = $\sqrt{(2-n)^2 + (2-2)^2 + (2-2)^2}$ So $\begin{cases} D_1 = \sqrt{1+1+1} & = \sqrt{3} = 1.73 \rightarrow \\ D_2 = \sqrt{1+1+0} = \sqrt{2} = 1.414 \rightarrow \\ D_3 = \sqrt{0+4+0} = \sqrt{4} = 2 \\ D_4 = \sqrt{0+4+1} = \sqrt{5} = 2.236 \end{cases}$ Dy = $\sqrt{0+4+1} = \sqrt{5} = 2.236$ Dy = $\sqrt{0+4+1} = \sqrt{5} = 2.449$ Dy = $\sqrt{0+4+1} = \sqrt{5} = 2.828$ Dy = $\sqrt{1+4+1} = \sqrt{5} = 2.828$ Dy = $\sqrt{1+4+1} = \sqrt{5} = 3$ C Dy = $\sqrt{1+1+1} = \sqrt{11} = 3.3166$ Dy = $\sqrt{1+1+0} = \sqrt{1} = 3.3166$ Dy = $\sqrt{1+1+0} = \sqrt{1} = 1.414$ Dy = $\sqrt{1+1+0} = \sqrt{1} = 1.414$	Class R: (17.2) (2.2.2) (1.2-1) (2.2.3)	) D8 (
Test Data = $(1,0,1)$ La distance = $\sqrt{(2-n)^2 + (2-2)^2 + (2-2)^2}$ So $\begin{cases} D_1 = \sqrt{1+1+1} & = \sqrt{3} = 1.73 \rightarrow \\ D_2 = \sqrt{1+1+0} = \sqrt{2} = 1.414 \rightarrow \\ D_3 = \sqrt{0+4+0} = \sqrt{4} = 2 \\ D_4 = \sqrt{0+4+1} = \sqrt{5} = 2.236 \end{cases}$ Dy = $\sqrt{0+4+1} = \sqrt{5} = 2.236$ Dy = $\sqrt{0+4+1} = \sqrt{5} = 2.449$ Dy = $\sqrt{0+4+1} = \sqrt{5} = 2.828$ Dy = $\sqrt{1+4+1} = \sqrt{5} = 2.828$ Dy = $\sqrt{1+4+1} = \sqrt{5} = 3$ C Dy = $\sqrt{1+1+1} = \sqrt{11} = 3.3166$ Dy = $\sqrt{1+1+0} = \sqrt{1} = 3.3166$ Dy = $\sqrt{1+1+0} = \sqrt{1} = 1.414$ Dy = $\sqrt{1+1+0} = \sqrt{1} = 1.414$	Class C: (-1, -1, -1) (0, -1, -2) (0, -1, 1) (-1, -	2, 1)012
Lo distance = $\sqrt{(h_2-n)^2+(y_1-y_1)^2+(z_2-z_1)^2}$ $(y_1-y_1)^2+(y_1-y_1$		-
Lo distance = $\sqrt{(h_2-n)^2+(y_1-y_1)^2+(z_2-z_1)^2}$ $(y_1-y_1)^2+(y_1-y_1$	Test Data = (1,0,1)	7
$\begin{cases} 0 & 0 & -\sqrt{1+1+1} & -\sqrt{3} & -\sqrt{1+1+0} \\ 0 & -\sqrt{1+1+0} & -\sqrt{2} & -\sqrt{1+1+0} \\ 0 & -\sqrt{1+1+0} & -\sqrt{1+1} & -\sqrt{2} \\ 0 & -\sqrt{1+1+0} & -\sqrt{1+1+0} \\ 0 & -\sqrt{1+1+0} & -\sqrt{1+0} \\ 0 & -\sqrt{1+0} & -1+0$	La distance = \( \beta_2 - n)^2 + (4 - y)^2 + (2_2 - 2_1)^2	
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$		
$D_{5} = \sqrt{0+4+1} = \sqrt{5} = 2.236$ $D_{6} = \sqrt{1+4+4} = \sqrt{6} = 2.449$ $D_{7} = \sqrt{0+4+4} = \sqrt{8} = 2.828$ $D_{8} = \sqrt{1+4+4} = \sqrt{9} = 3$ $D_{9} = \sqrt{4+1+4} = \sqrt{9} = 3$ $D_{10} = \sqrt{1+1+4} = \sqrt{11} = 3.3166$ $D_{11} = \sqrt{1+1+0} = \sqrt{2} = 1.414$ $D_{12} = \sqrt{2} + 4+6 = \sqrt{9} = 2.8284$	€0 (D= √1+1+1 = √3 = 1.73 ->	-
$D_{5} = \sqrt{0+4+1} = \sqrt{5} = 2.236$ $D_{6} = \sqrt{1+4+4} = \sqrt{6} = 2.449$ $D_{7} = \sqrt{0+4+4} = \sqrt{8} = 2.828$ $D_{8} = \sqrt{1+4+4} = \sqrt{9} = 3$ $D_{9} = \sqrt{4+1+4} = \sqrt{9} = 3$ $D_{10} = \sqrt{1+1+4} = \sqrt{11} = 3.3166$ $D_{11} = \sqrt{1+1+0} = \sqrt{2} = 1.414$ $D_{12} = \sqrt{2} + 4+6 = \sqrt{9} = 2.8284$	De 2 11+1+0 = 12 - 11-114 - 11-114 - 2	
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	$D_1 = \sqrt{0+4+1} = \sqrt{5} = 2.236$	The second secon
$\begin{array}{cccccccccccccccccccccccccccccccccccc$		
$\begin{array}{c} D_{8} = \sqrt{1+4+4} = 19 = 3 \\ 0 = \sqrt{1+1+4} = \sqrt{9} = 3 \\ 0 = \sqrt{1+1+9} = \sqrt{11} = 3.3166 \\ 0 = \sqrt{1+1+0} = \sqrt{2} = 1.414 \\ 0 = \sqrt{2} = \sqrt{2} + 4 + 6 = \sqrt{9} = 2.8284 \end{array}$	Dg = V 0 + 4 + 1 = 1 3 = 2 : 14 9	
$\begin{array}{c} D_{8} = \sqrt{1+4+4} = 19 = 3 \\ 0 = \sqrt{1+1+4} = \sqrt{9} = 3 \\ 0 = \sqrt{1+1+9} = \sqrt{11} = 3.3166 \\ 0 = \sqrt{1+1+0} = \sqrt{2} = 1.414 \\ 0 = \sqrt{2} = \sqrt{2} + 4 + 6 = \sqrt{9} = 2.8284 \end{array}$	D7= 10+44 - 18 = 2.828	
$\begin{array}{c} 0g = \sqrt{9+1+4} = \sqrt{9} = 3 \\ 0 = \sqrt{1+1+3} = \sqrt{11} = 3.3166 \\ 0 = \sqrt{1+1+0} = \sqrt{2} = 1.414 \\ 0 = \sqrt{2} = \sqrt{2} + 4 + 6 = \sqrt{9} = 2.8284 \end{array}$	D8 = 51+4+4 = 50 = 3	
D10 = VI+1+3 = VII = 3.3166  D11 = VI+1+0 = V2 = 1.414 ->  D12 = V2+4+0 = V8 = 2.8284		-
D12= 17+4+0 = 12 = 1,414 -> D12= 17+4+0 = 18 = 2.8284	Dg = 14144 = 11 = 32166	-
	Du = VI+1+0 = V2 = 1,414 ->	6
	D12= 17:44 to -19 = 2.8284	
		6 2
		-

#### 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:**

result=[]

```
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):
```

# compute L2 distance for all test and train samples

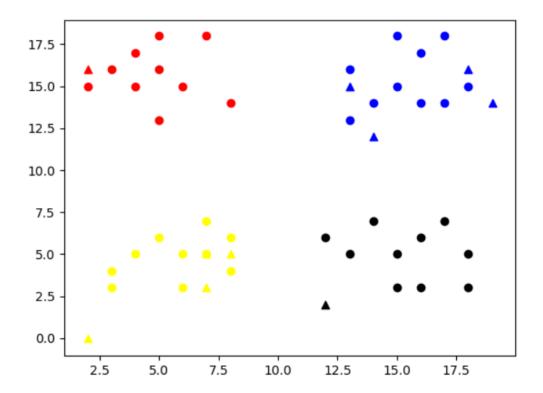
```
distances = []
  for item1 in newInput:
        d = []
        for item2 in dataSet:
                distance = np.sqrt(np.sum((item1-item2)**2)) #calclulating L2 distance
                d.append(distance)
        distances.append(d)
  print(distances)
  # decide which class the test samples belong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

```
def kNNClassify(newInput, dataSet, labels, k):
     result=[]
     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)
     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
     x_train, y_train, x_test, y_test = load()
 12 x_train = x_train.reshape(60000,28,28)
 13 x \text{ test} = x \text{ test.reshape}(10000, 28, 28)
 14 x train = x train.astype(float)
     x_test = x_test.astype(float)
 16 ▼ def kNNClassify(newInput, dataSet, labels, k):
 17
          result=[]
          # compute L2 distance for all test and train samples
 21
          distances = np.zeros((len(newInput), len(dataSet)))
 22 ▼
          for i in range(len(newInput)):
              for j in range(len(dataSet)):
 23 ▼
 24
                  distance = np.sqrt(np.sum((newInput[i]-dataSet[j])**2))
 25
                  distances[i, j] = distance
 27
 28 ▼
          for i in range(len(newInput)):
 29
              label value = np.zeros(10)
              knn indices = np.argsort(distances[i])[:k]
 31 ▼
              for j in range(len(knn_indices)):
                  label = labels[knn indices[j]]
 32
                  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))
 41
     print ("---classification accuracy for knn on mnist: %s ---" %result)
 42
     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>
```