Riken Patel

02/13/2021

ECE 407 Homework 3

The dictionary of classes with the total number of images in that class:

>> 0 4932

>> 1 5678

>> 2 4968

>> 3 5101

>> 4 4859

>> 5 4506

>> 6 4951

>> 7 5175

>> 8 4842

>> 9 4988

>>

>> Total number of images per class:

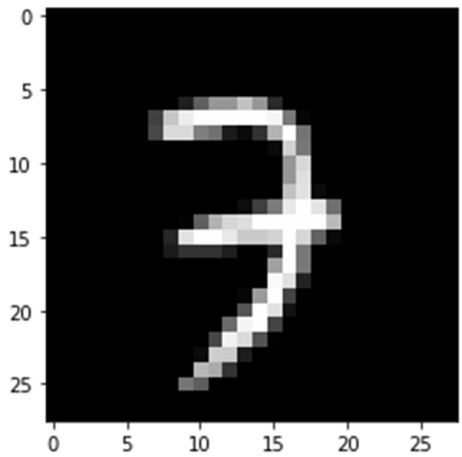
>> [4932, 5678, 4968, 5101, 4859, 4506, 4951, 5175, 4842, 4988]

>>

>> Expected: 9

>> Answer: 7

>>



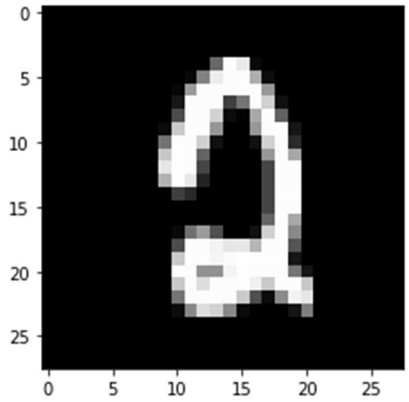
>> The prosterior probabilities for each of the 10 classes are: [-321.30528320149745, -450.4801838134944, -311.65059713067217, -294.80830912781374, -265.49101479083623, -263.34754797446953, -313.65335847776976, -250.87658377320534, -269.96866238492913, -237.42323411023528]

>>

>> Expected: 7

>> Answer: 2

>>



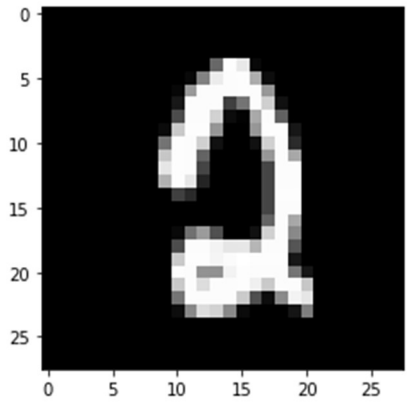
>> The prosterior probabilities for each of the 10 classes are: [-391.18094804183664, -471.2288285470782, -313.6144570014757, -382.81712926327555, -310.3973444767139, -356.6470911686817, -335.7274645308598, -279.20068994327414, -299.52018992456647, -320.4186873571524]

>>

>> Expected: 3

>> Answer: 2

>>



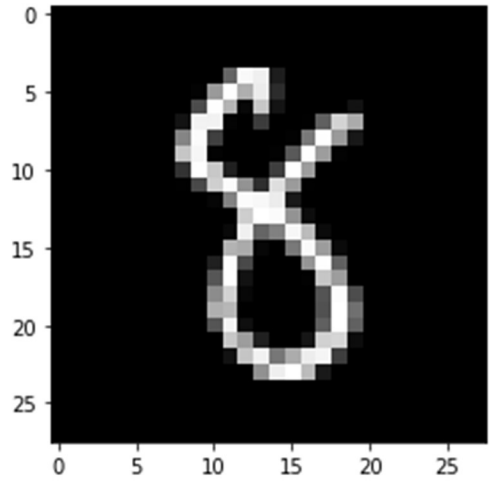
>> The prosterior probabilities for each of the 10 classes are: [-357.43932398463966, -271.6475814664299, -237.34734870454102, -234.52349060538887, -273.5695267510583, -251.25034631412566, -261.8959840103364, -269.46101233031766, -275.17676436840986, -291.8252044888843]

>>

>> Expected: 3

>> Answer: 8

>>



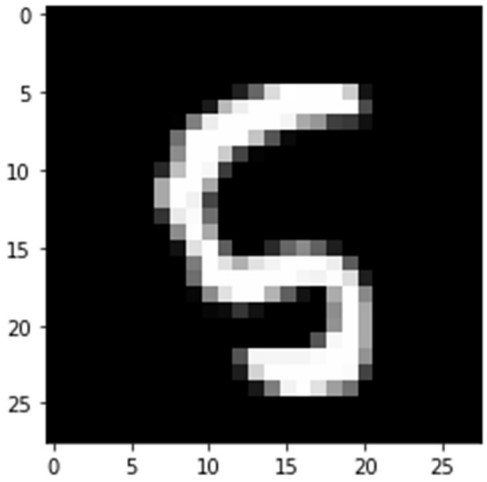
>> The prosterior probabilities for each of the 10 classes are: [-313.0016371623362, -602.6551512713531, -308.2768284646239, -321.57711429207467, -451.1205295664968, -364.6162443232696, -325.81083100811117, -507.1684846112748, -417.7757641815645, -501.07320673256794]

>>

>> Expected: 3

>> Answer: 5

>>



>> The prosterior probabilities for each of the 10 classes are: [-325.66032484576294, -335.05975700908203, -281.77257386032545, -218.15644274290088, -339.9756198277668, -225.27849737120704, -280.44053036564395, -378.0952016819456, -222.48588985966782, -315.21337493248296]

>> Accuracy Rate: 84%

>> Error Rate on Mnist test set: 16%

CODE:

```
# Riken Patel
# ECE 407 Homework 3
# 02/13/2021

import numpy as np
from keras.datasets import mnist
import matplotlib.pyplot as plt
import random
import math

(imgTrain, labelTrain), (imgTest,labelTest) = mnist.load_data() #loads the mnist
data using keras

#Separating the mnist data:
imgTrain = imgTrain[:50000]
labelTrain = labelTrain[:50000]
imgTest = imgTest[:10000]
labelTest = labelTest[:10000]

#Creating a dictionary of arrays to store the different classes:
trainingSet = {'0':[],'1':[],'2':[],'3':[],'4':[],'5':[],'6':[],'7':[],'8':[],'9'
:[]} #empty arrays to be appended
```

```
probabilitySet = {'0':[],'1':[],'2':[],'3':[],'4':[],'5':[],'6':[],'7':[],'8':[],
'9':[]} #empty arrays to be appended
totalClass = [0,0,0,0,0,0,0,0,0,0] #number of certain class images array

#Thresholding for the image converting the image to binary
for i in range(50000):
    ind = labelTrain[i] #searching for the label
    img = np.reshape(imgTrain[i],784) #reshaping the 28x28 array to 1x784 array f
or faster calculations
    for col in range(len(img)):
            if img[col] >= 180: #thresholding for a value of 180
                img[col] = 1
            else:
                img[col] = 0
    trainingSet[str(ind)].append(img) #pushing the image into it's particular key
/class
    totalClass[ind] += 1

#Checking if the tally is correct:
print("The dictionary of classes with the total number of images in that class: "
)
for key, value in trainingSet.items():
    print(key, len(value))

#Counting the total images in each class:
print("\n Total number of images per class: ")
print(totalClass)

pClass = np.zeros(10) #array to store class distribution.
xClass = np.zeros(10) #array to store class distribution.
xCalc = np.zeros(784) #array of zeros to input the success rate of 1 happening fo
r the 784 pixels.
pCalc = np.zeros(784) #array of zeros to input the success rate of 1 happening fo
r the 784 pixels.

#For each pixel in each image:
for key, value in trainingSet.items(): #iterates through the key of the dictionar
y for the values in the training set
    numImgs = len(trainingSet[str(key)]) #number of images in each class
    for i in range(784): #for each pixel in each image
        counter1 = 0 #counts the 1's
        counter0 = 0 #counts the 0's
        for img in range(numImgs):
            if trainingSet[str(key)][img][i] == 1: #focuses on the first image of
 each pixel
```

```
                counter1 += 1
            elif trainingSet[str(key)][img][i] == 0:
                counter0 += 1
        probCalc = counter1/numImgs #calculates the probabilty value
        if probCalc == 0 or probCalc == 1:
            probCalc = (counter1 + 1)/(numImgs + 2) #Laplace smoothing to not get
 extreme data of either 0 or 1, but somewhere in the middle.
        x_Calc = counter0/numImgs
        if x_Calc == 0 or  x_Calc == 1:
            x_Calc = (counter0 + 1)/(numImgs + 2) #Laplace smoothing again to avo
id extremes of 0 or 1.
        pCalc[i] = probCalc #storing into pCalc array
        xCalc[i] = x_Calc #storing into xCalc array
        probabilitySet[str(key)].append(pCalc[i]) #storing the probabilities of c
lass and pixel in probabilitySet

#Creating empty matrices for the misclassified test digits:
misclassifiedImg = []
misclassifiedProb = []
misclassifiedRes = []
misclassifiedLabel = []

#Test digits:
for i in range(len(imgTest)):
    maxProb = [] #empty array of maxProb that will be appended.
    for num in range(10):
        sumTest = 0 #resets sumTest to 0 after each iteration
        probFinal = 0 #resets probFinal to 0 after each iteration
        img = imgTest[i].reshape(1,784)
        for j in range(784):
            #calculating the total probability by using Bernoulli distribution lo
g formula:
            sumTest += (img[0][j]/255)*math.log2(probabilitySet[str(num)][j])+(1-
(img[0][j]/255))*math.log2(1-probabilitySet[str(num)][j])
        classProb = len(trainingSet[str(labelTrain[num])])/60000 #calculating the
 probability of each image being in that class
        probFinal = sumTest + math.log2(classProb) #calculating final probability
        maxProb.append(probFinal) #appending the probFinal results into maxProb e
mpty array
    maxAnswer = np.max(maxProb) #finding the maximum value in maxProb array.
    res = maxProb.index(maxAnswer) #finding the index of maxAnswer of maxProb
    if res != labelTest[i]: #putting the mismatched images/labels/results into a
separate array to find the accuracyRate
            misclassifiedImg.append(imgTest[i])
            misclassifiedProb.append(maxProb)
```

```
            misclassifiedRes.append(res)
            misclassifiedLabel.append(labelTest[i])

#Five different test digits:
for i in range(5):
    print("\n")
    randInd = random.randint(1, len(misclassifiedImg))
    print("Expected: " + str(misclassifiedRes[randInd]))
    print("Answer: " + str(misclassifiedLabel[randInd]))
    print("\n")
    print("The prosterior probabilities for each of the 10 classes are: " + str(m
isclassifiedProb[randInd]))
    print("\n")
    plt.figure()
    plt.imshow(misclassifiedImg[randInd].reshape(28,28),cmap = 'gray')
accuracyRate = math.floor(((10000-
len(misclassifiedImg))/(10000))*100) #calculating the accuracy rate by taking the
 amount of images in test - length mismatched images / amount of test images mult
iplied by 100 for %
errorRate = 100-(accuracyRate)
print("Accuracy Rate: " + str(math.floor(accuracyRate)) +"%")
print("Error Rate on Mnist test set: " + str(errorRate) + "%")
```