Aaron P. Mills Z-23547104

Dr. Ghoraani CAP 4613 Intro to Deep Learning

Assignment 2 6 February 2022

https://colab.research.google.com/drive/1FfbrSe5vSjtYv6rie2zYQanqzIvyS12D?usp=sharing

Note: The code PDF is below, after the handwork solutions.

Problem 1)

a) Answer: ↓

	Fraudulent	Non-Fraudulent
Fraudulent	19	6
Non-Fraudulent	6	9

b) Answer: 70%

$$Model\ Accuracy = \frac{TP+TN}{\#\ of\ samples} = \frac{19+9}{19+6+6+9} = \frac{28}{40} = 0.7 = 70\%$$

c) Answer: 76%

$$Sensitivity = \frac{TP}{TP+FN} = \frac{19}{19+6} = \frac{19}{25} = 0.76 = 76\%$$

d) Answer: 60%

$$Specificty = \frac{TN}{TN+FP} = \frac{9}{9+6} = \frac{9}{15} = 0.6 = 60\%$$

e) Answer: 76%

$$F1\ Score = 2 * \frac{sensitivity*precision}{sensitivity+precision} = 2 * \frac{0.76*0.76}{0.76+0.76} = 0.76 = 76\%$$

$$Precision = \frac{TP}{TP+FP} = \frac{19}{19+6} = \frac{19}{25} = 0.76 = 76\%$$

f) Answer: Recall = 76%, precision = 76%

$$Recall = sensitivity = \frac{TP}{TP+FN} = \frac{19}{19+6} = \frac{19}{25} = 0.76 = 76\%$$

$$Precision = \frac{TP}{TP+FP} = \frac{19}{19+6} = \frac{19}{25} = 0.76 = 76\%$$

PDF-XChange Editor Click to BUY NOW! www.tracker-software.com

Problem 2)

a) Answer: ↓

	Class 1	Class 0
Class 1	7	2
Class 0	3	5

b) Answer: $Accuracy \approx 70.59\%, sensitivity = 77.78\%, specificity \approx 62.5\%$

$$Accuracy = \frac{\sum diagonal}{\#\ of\ samples} = \frac{TP+TN}{TP+TN+FP+FN} = \frac{7+5}{7+5+3+2} = \frac{12}{17} \approx 70.59\%$$

$$Sensitivity = \frac{TP}{TP+FN} = \frac{7}{7+2} = \frac{7}{9} \approx 77.78\%$$

$$Specificity = \frac{TN}{TN+FP} = \frac{5}{5+3} = \frac{5}{8} = 62.50\%$$

c) Answer: $w_0 = -1, w_1 = \frac{2}{3}, w_2 = \frac{1}{3}$

$v = w_1x_1 + w_2x_2 + w_0 = 0$

$0 = w_1(1.5) + w_2(0) + w_0$ • Point 1: (1.5,0);

$0 = w_1(0) + w_2(3) + w_0$ • Point 2: (0,3)

$0 = w_1(1.5) - 1 \rightarrow 1 = w_1(1.5) \rightarrow w_1 = \frac{1}{1.5} = \frac{2}{3}$ • let $w_0 = -1$

$0 = w_2(3) - 1 \rightarrow 1 = w_2(3) \rightarrow w_2 = \frac{1}{3}$ • ↑

d) Answer: New Sample 1: $v = \frac{2}{3}, y = 1, Class\ Label = Class\ 1$;

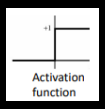
New Sample 2: $v = -\frac{7}{3}, y = 0, Class\ Label = Class\ 0$

$v = \frac{2}{3}x_1 + \frac{1}{3}x_2 - 1$ • v local field found in part (c)

$S_{i+1}.v = \frac{2}{3}(2) + \frac{1}{3}(-1) - 1 = \frac{2}{3} \approx 0.67$ •new sample 1's v

$S_{i+2}.v = \frac{2}{3}(-0.5) + \frac{1}{3}(0.5) - 1 = -\frac{7}{6} \approx -1.17$ • new sample 2's v

$$y = \begin{cases} 1 & if\ v \geq 0 \\ 0 & if\ v < 0 \end{cases}$$

- based on 

$S_{i+1}.y = 1$

- $S_{i+1}.v = \frac{2}{3} \geq 0 \rightarrow y = 1$

$S_{i+2}.y = 0$

- $S_{i+2}.v = -\frac{7}{6} < 0 \rightarrow y = 0$

Predicted Classification Label = {Class 1} if y == 1, else {Class 0} if y == 0

$S_{i+1}.Classification\ Label = Class\ 1$

- y = 1

$S_{i+2}.Classification\ Label = Class\ 0$

- y = 0

Aaron P. Mills / Z-23547104 / Dr. Ghoraani

Intro to Deep Learning / CAP 4613 / Assignment 2 / 6 February 2022

https://colab.research.google.com/drive/1FfbrSe5vSjtYv6rie2zYQanqzIvyS12D?usp=sharing

```
# Aaron P. Mills /               Z-23547104 /
# Dr. Ghoraani
# Intro to Deep Learning  /       CAP 4613 /
# Assignment 2             /      6 February 2022
# Discription: In this assignment, I collect data, organize it into sets, display random s
# https://colab.research.google.com/drive/1FfbrSe5vSjtYv6rie2zYQanqzIvyS12D?usp=sharing
##############################################################################################
#header block   - includes heading, imports, functions, classes, ect.
import math as mth
import numpy as np
import matplotlib.pyplot as plt
from keras.datasets import mnist

#Problem 3)
#part b) write function which takes set of (image & label) as input and displays a figure
def plot_numb(imgs=[0],labels=[0]):         #takes set of images & corresponding label
    plt.figure()                            #figure is the box that holds images, so it be
    for i in range( len(imgs) ):            #to print all images
        plt.subplot(2,5,i+1)                #2 rows, 5 cols, i+1 position in grid matrix
        plt.imshow(imgs[i],cmap='gray')     #needed for each image display
        plt.title('Label: '+ str(labels[i]))#use corresponding label in tittle name
    plt.show()                              #display entire figure
```

```
#part a) obtaining, splitting, and printing data from http://yann.lecun.com/exdb/mnist/
(x_train_all,y_train_all),(x_test_all,y_test_all) = mnist.load_data()
print(f"There exist {x_train_all.shape[0]} images of size {x_train_all.shape[1]}x{x_train_
print(f"There exist {x_test_all.shape[0]} images of size {x_test_all.shape[1]}x{x_test_all

#part b) write function which takes set of (image & label) as input and displays a figure
#NOTE: part b) may be found in the header (first code) block, because it is a function

#testing plot_numb; this is NOT a requirement, so I commented it out
# print("Testing plot_numb() function*****************************************************
# xlist = []                                          #list for images
# ylist = []                                          #list for labels
# for digit in range(10):                             #10 images from 0-9
#     x_train_d = x_train_all[y_train_all==digit,:,:] #set of imgs with label==digit
#     x_train_i = x_train_d[0,:,:]                    #first img with label==digit
#     y_train_d = y_train_all[y_train_all==digit]     #corresponding set of labels to img;
#     y_train_i = y_train_d[0]                        #corresponding label to img; first l
#     xlist.append(x_train_i)                         #add to list of imgs
#     ylist.append(y_train_i)                         #add to corresponding list of labels
# plot_numb(xlist,ylist)                            #plot all imgs from 0-9

```

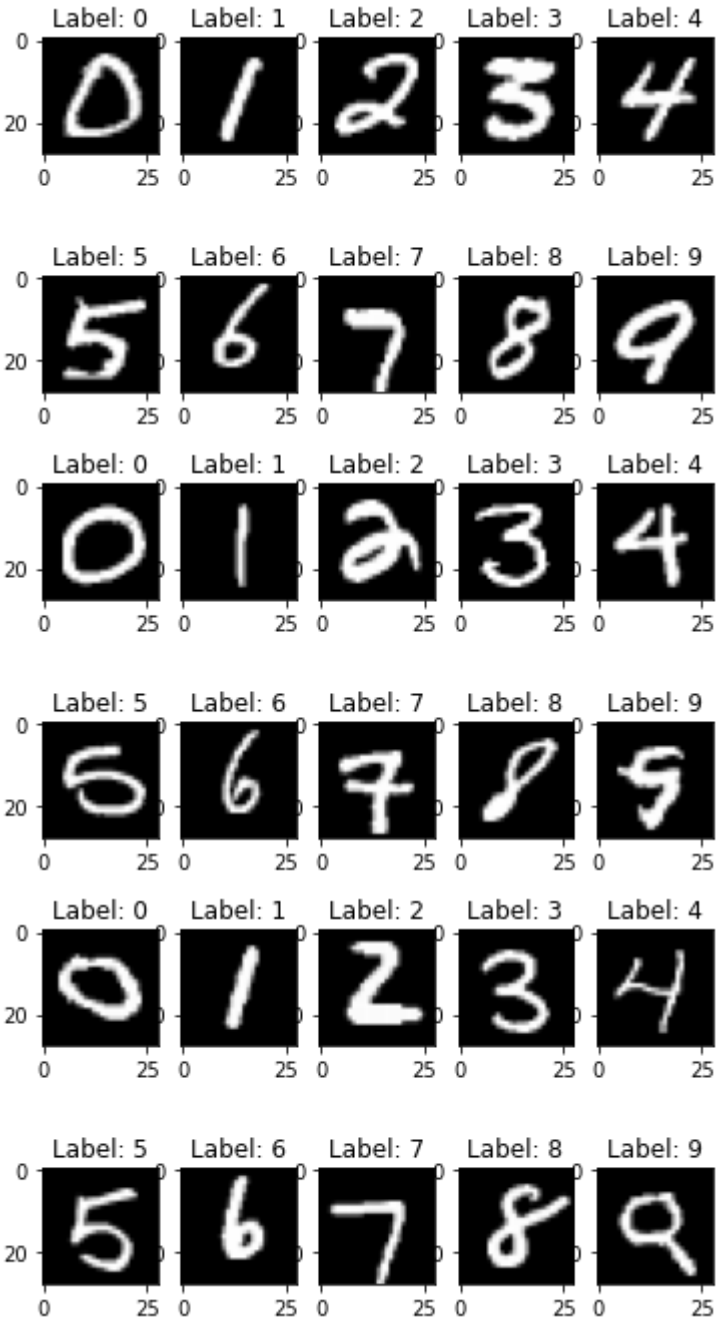
```
#part c) create loop to call plot function from part b) for 3 times, and ensure eac  image
for i in range(3):
    img_list = []
    label_list = []
    for digit in range(10):
        x_train_d   = x_train_all[y_train_all==digit]   #matrix of ONLY imgs that are of c
        y_train_d   = y_train_all[y_train_all==digit]   #corresponding label for digits
        num_imgs_d  = x_train_d.shape[0]                #get number of image of digit
        train_ind_d  = np.arange(0,num_imgs_d)          #use that number in creating indic
        train_ind_ds = np.random.permutation(train_ind_d)#randomize the order of the indic
        x_train_dshuffle = x_train_d[train_ind_ds,:,:]  #shuffled data = original_data@[ra
        y_train_dshuffle = y_train_d[train_ind_ds]      #corresponding label for shuffled
        ind_ds   = np.random.randint(0,num_imgs_d)      #randomized index for more randomi
        x_train_dimg = x_train_dshuffle[ind_ds,:,:]     #randomized img of digit = shuffle
        y_train_dimg = y_train_dshuffle[ind_ds]         #corresponding label for such imag
        img_list.append(x_train_dimg)                   #add random img of digit to list
        label_list.append(y_train_dimg)                 #and its label
    plot_numb(img_list,label_list)                      #display each random img of all di

#part d) split the training data so that a random 20% is allocated to validation, while th
num_train_imgs  = x_train_all.shape[0]                  #get number of training images
train_ind_all   = np.arange(0,num_train_imgs)           #get indices representing entire s
train_ind_all_s = np.random.permutation(train_ind_all)  #randomized set indices
x_train_all_s   = x_train_all[train_ind_all_s,:,:]      #randomized_set = data@[randomized
y_train_all_s   = y_train_all[train_ind_all_s]          #^with corresponding labels

x_valid = x_train_all_s[0:int(0.20*num_train_imgs),:,:] #validity set is 20% of random set
y_valid = y_train_all_s[0:int(0.20*num_train_imgs)]     #same for label
x_train = x_train_all_s[(int(0.20*num_train_imgs)):num_train_imgs,:,:]  #training set is t
y_train = y_train_all_s[(int(0.20*num_train_imgs)):num_train_imgs]      #same for label

print("|Number of Samples per Set|")
print(f"x_valid: {x_valid.shape[0]}\ty_valid: {y_valid.shape[0]}")      #shape returns [ca
print(f"x_train: {x_train.shape[0]}\ty_train: {y_train.shape[0]}")      #^
print(f"The validation set holds {x_valid.shape[0]} images, while training holds {x_train.
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
11501568/11490434 [==============================] - 0s 0us/step
There exist 60000 images of size 28x28 in the training set.
There exist 10000 images of size 28x28 in the testing set.
```




```
|Number of Samples per Set|
x_valid: 12000  y_valid: 12000
x_train: 48000  y_train: 48000
The validation set holds 12000 images, while training holds 48000
```

✓ 9s completed at 5:34 AM