CPS4893AS3

Peiyan Wang 1129627

June 2022

**Contents**

Question 1 -------------------------------------------------------------------------------------------- 2

1. Image of code --------------------------------------------------------------------------------- 2

2. Code ---------------------------------------------------------------------------------------------- 3

3. Test ----------------------------------------------------------------------------------------------- 4

Question 2 -------------------------------------------------------------------------------------------- 5

Question 3 -------------------------------------------------------------------------------------------- 5

1. Image of code --------------------------------------------------------------------------------- 5

2. Code ----------------------------------------------------------------------------------------------6

3. Test ----------------------------------------------------------------------------------------------- 7

Question 4 -------------------------------------------------------------------------------------------- 8

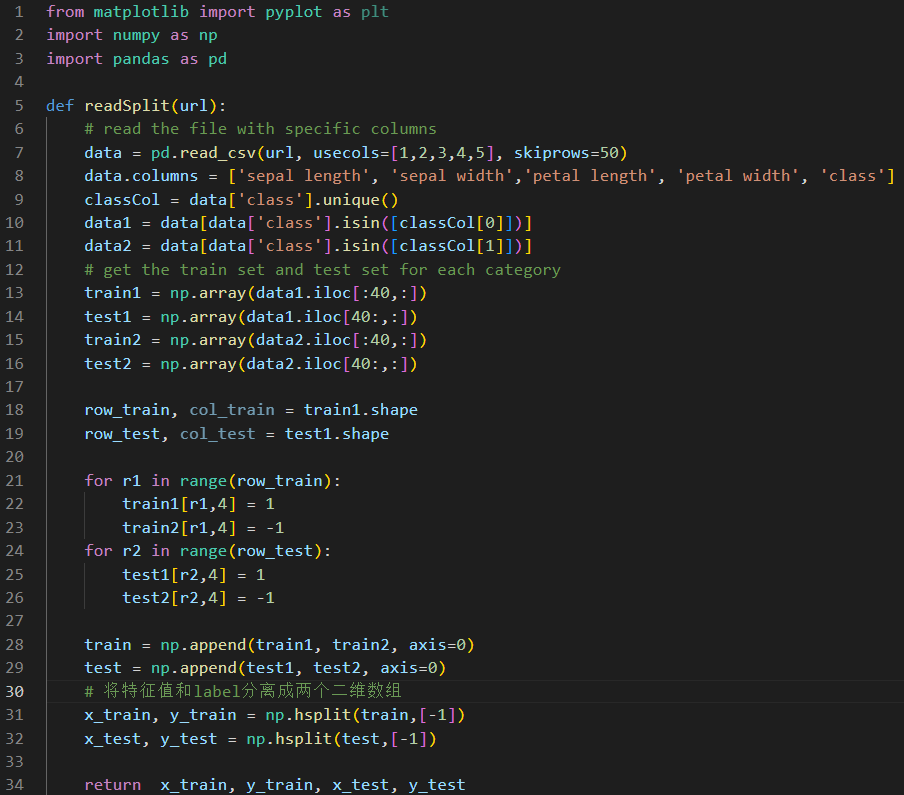
1. Image of code --------------------------------------------------------------------------------- 8

2. Code ----------------------------------------------------------------------------------------------9

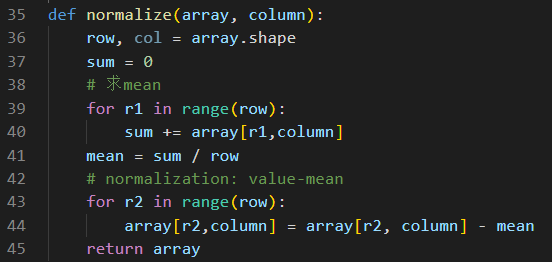
3. Test ----------------------------------------------------------------------------------------------- 10

**Question 1**

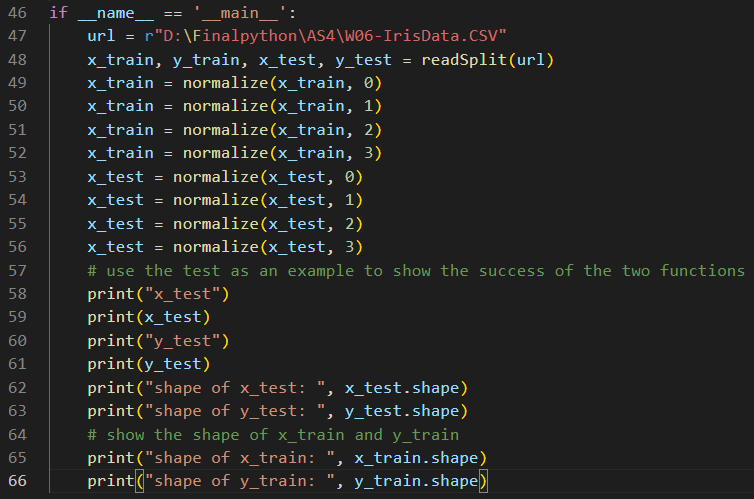
1. **Image of code for the better reading experience**
2. Function to read the file, split into train and test dataset, change the string value of label into integer label (1 or -1), and split the character values and label into two two-dimension array.



1. Function to normalize the train and test sets.



1. Main function to test the function



1. **Code details**

from matplotlib import pyplot as plt

import numpy as np

import pandas as pd

def readSplit(url):

# read the file with specific columns

data = pd.read\_csv(url, usecols=[1,2,3,4,5], skiprows=50)

data.columns = ['sepal length', 'sepal width','petal length', 'petal width', 'class']

classCol = data['class'].unique()

data1 = data[data['class'].isin([classCol[0]])]

data2 = data[data['class'].isin([classCol[1]])]

# get the train set and test set for each category

train1 = np.array(data1.iloc[:40,:])

test1 = np.array(data1.iloc[40:,:])

train2 = np.array(data2.iloc[:40,:])

test2 = np.array(data2.iloc[40:,:])

row\_train, col\_train = train1.shape

row\_test, col\_test = test1.shape

for r1 in range(row\_train):

train1[r1,4] = 1

train2[r1,4] = -1

for r2 in range(row\_test):

test1[r2,4] = 1

test2[r2,4] = -1

train = np.append(train1, train2, axis=0)

test = np.append(test1, test2, axis=0)

# 将特征值和label分离成两个二维数组

x\_train, y\_train = np.hsplit(train,[-1])

x\_test, y\_test = np.hsplit(test,[-1])

return x\_train, y\_train, x\_test, y\_test

def normalize(array, column):

row, col = array.shape

sum = 0

# 求mean

for r1 in range(row):

sum += array[r1,column]

mean = sum / row

# normalization: value-mean

for r2 in range(row):

array[r2,column] = array[r2, column] - mean

return array

if \_\_name\_\_ == '\_\_main\_\_':

url = r"D:\Finalpython\AS4\W06-IrisData.CSV"

x\_train, y\_train, x\_test, y\_test = readSplit(url)

x\_train = normalize(x\_train, 0)

x\_train = normalize(x\_train, 1)

x\_train = normalize(x\_train, 2)

x\_train = normalize(x\_train, 3)

x\_test = normalize(x\_test, 0)

x\_test = normalize(x\_test, 1)

x\_test = normalize(x\_test, 2)

x\_test = normalize(x\_test, 3)

# use the test as an example to show the success of the two functions

print("x\_test")

print(x\_test)

print("y\_test")

print(y\_test)

print("shape of x\_test: ", x\_test.shape)

print("shape of y\_test: ", y\_test.shape)

# show the shape of x\_train and y\_train

print("shape of x\_train: ", x\_train.shape)

print("shape of y\_train: ", y\_train.shape)

1. **Test**

Since the contents in the train sets (both contains character values and label value) are a lot, the result will first show the contents of test sets (both contains character values and label value respectively) and the shape of each test sets. Then, the result will show the shape of each train sets.

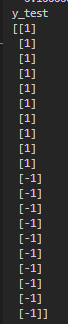
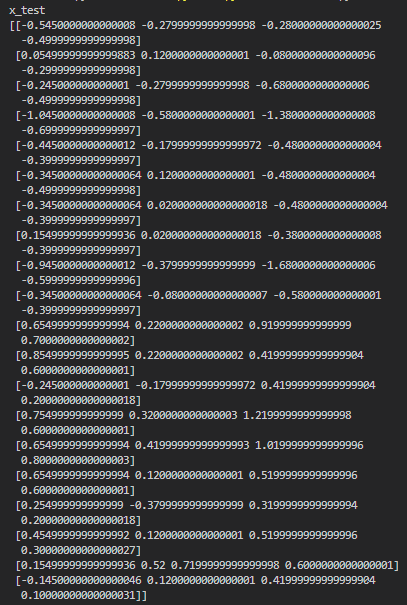


Figure 1: Test sets after normalization by mean

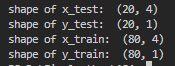


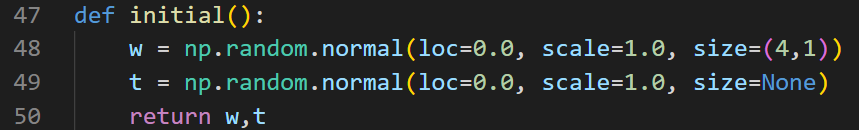
Figure 2: Shape of each train sets and test sets

**Question 2**

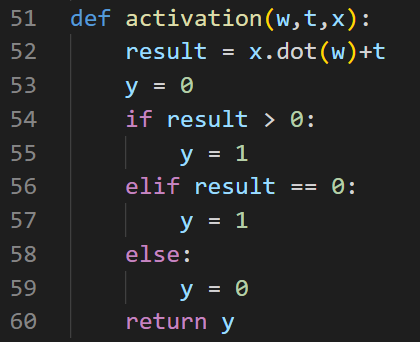
1. Input dimension is 4 for both train set and test set, since there are four characters.
2. Output dimension is 1 for both train set and test set, since there is one value representing the output, -1 or 1.
3. The weight number is 4 since there are four characters.
4. The bias number is 1 since the layer number is 1.

**Question 3**

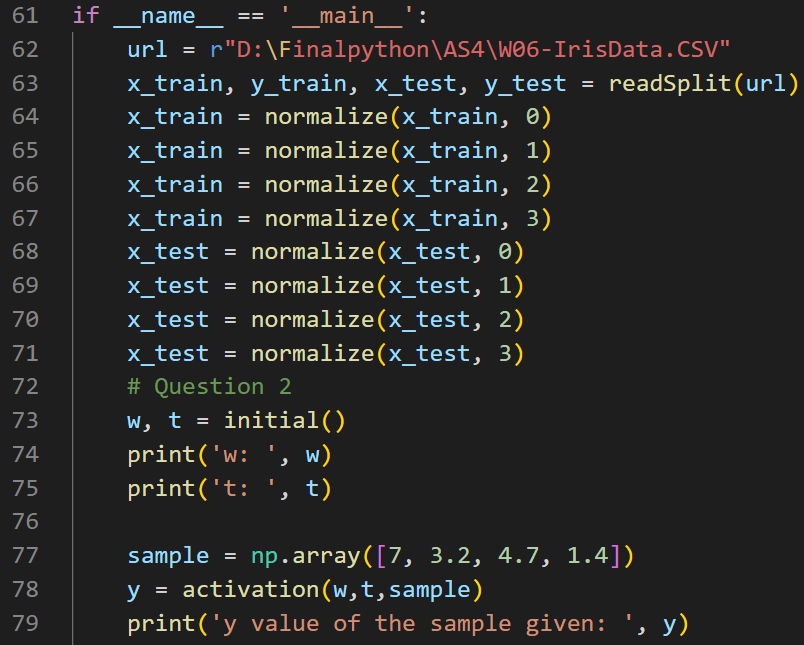
1. **Image of code for the better reading experience**
2. Function to initialize the values of w and t, which are sampled from the standard normal distribution with 0 mean and 1 std



1. Function of activation



1. Main function to test the two functions above



1. **Code details**

def initial():

w = np.random.normal(loc=0.0, scale=1.0, size=(4,1))

t = np.random.normal(loc=0.0, scale=1.0, size=None)

return w,t

def activation(w,t,x):

result = x.dot(w)+t

y = 0

if result > 0:

y = 1

elif result == 0:

y = 1

else:

y = 0

return y

if \_\_name\_\_ == '\_\_main\_\_':

url = r"D:\Finalpython\AS4\W06-IrisData.CSV"

x\_train, y\_train, x\_test, y\_test = readSplit(url)

x\_train = normalize(x\_train, 0)

x\_train = normalize(x\_train, 1)

x\_train = normalize(x\_train, 2)

x\_train = normalize(x\_train, 3)

x\_test = normalize(x\_test, 0)

x\_test = normalize(x\_test, 1)

x\_test = normalize(x\_test, 2)

x\_test = normalize(x\_test, 3)

# Question 2

w, t = initial()

print('w: ', w)

print('t: ', t)

sample = np.array([7, 3.2, 4.7, 1.4])

y = activation(w,t,sample)

print('y value of the sample given: ', y)

1. **Test**
2. Result of testing two functions above,

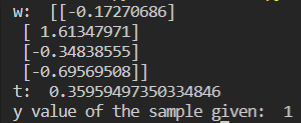


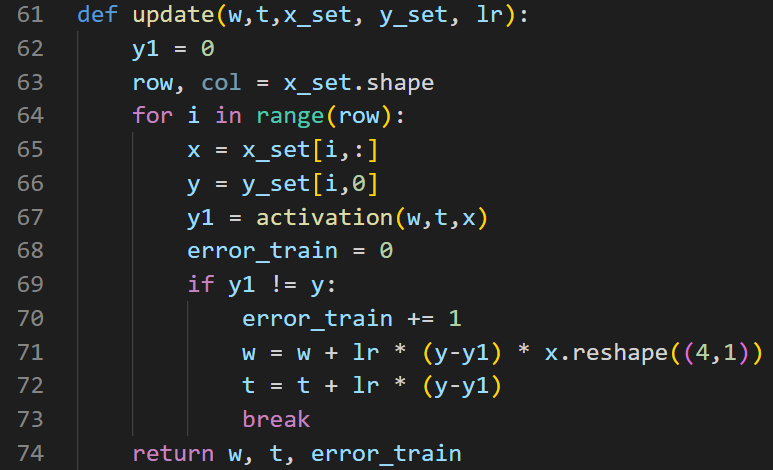
Figure 3: Classification errors results

*Mention: The code should be run together with the two functions shown in Question 1.*

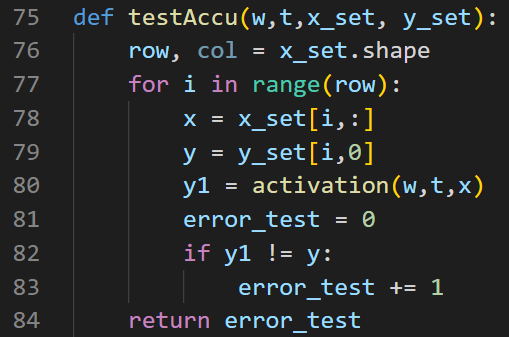
1. If initializing both w and t as 0, the algorithm which we use to update the value of w and t will lose it function. That is, the program cannot classify.

**Question 4**

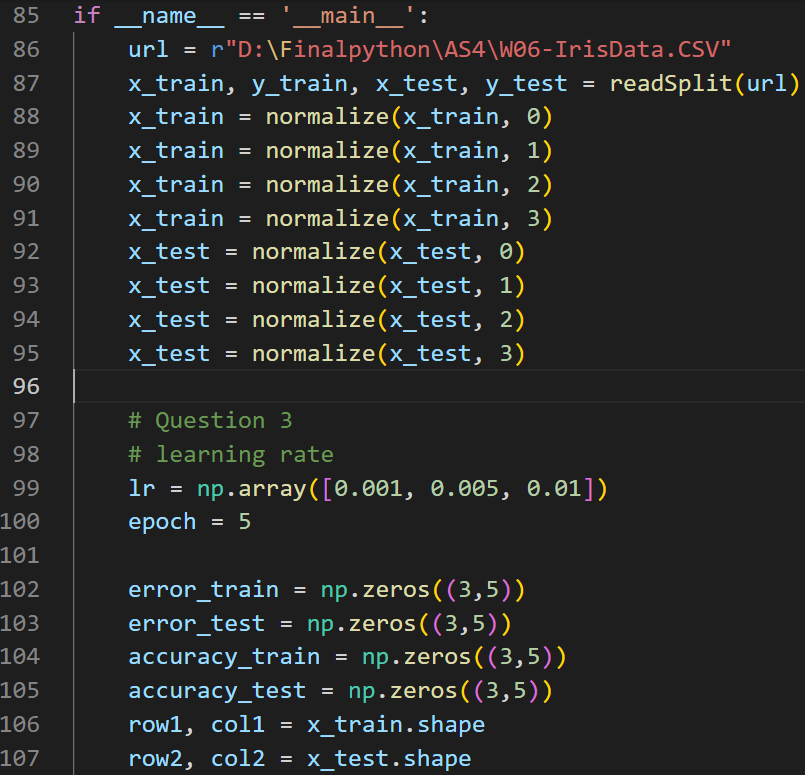
1. **Image of code for the better reading experience**
2. Function to update the value of w and t

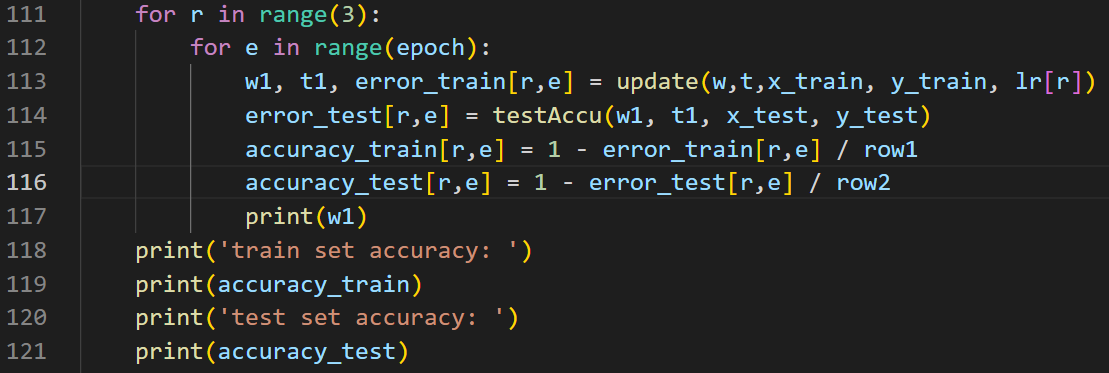


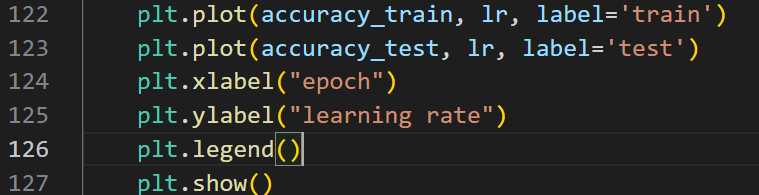
1. Function to calculate the accuracy of test set



1. Main function to test







1. **Code details**

def update(w,t,x\_set, y\_set, lr):

y1 = 0

row, col = x\_set.shape

for i in range(row):

x = x\_set[i,:]

y = y\_set[i,0]

y1 = activation(w,t,x)

error\_train = 0

if y1 != y:

error\_train += 1

w = w + lr \* (y-y1) \* x.reshape((4,1))

t = t + lr \* (y-y1)

break

return w, t, error\_train

def testAccu(w,t,x\_set, y\_set):

row, col = x\_set.shape

for i in range(row):

x = x\_set[i,:]

y = y\_set[i,0]

y1 = activation(w,t,x)

error\_test = 0

if y1 != y:

error\_test += 1

return error\_test

if \_\_name\_\_ == '\_\_main\_\_':

url = r"D:\Finalpython\AS4\W06-IrisData.CSV"

x\_train, y\_train, x\_test, y\_test = readSplit(url)

x\_train = normalize(x\_train, 0)

x\_train = normalize(x\_train, 1)

x\_train = normalize(x\_train, 2)

x\_train = normalize(x\_train, 3)

x\_test = normalize(x\_test, 0)

x\_test = normalize(x\_test, 1)

x\_test = normalize(x\_test, 2)

x\_test = normalize(x\_test, 3)

# Question 3

# learning rate

lr = np.array([0.001, 0.005, 0.01])

epoch = 5

error\_train = np.zeros((3,5))

error\_test = np.zeros((3,5))

accuracy\_train = np.zeros((3,5))

accuracy\_test = np.zeros((3,5))

row1, col1 = x\_train.shape

row2, col2 = x\_test.shape

for r in range(3):

for e in range(epoch):

w1, t1, error\_train[r,e] = update(w,t,x\_train, y\_train, lr[r])

error\_test[r,e] = testAccu(w1, t1, x\_test, y\_test)

accuracy\_train[r,e] = 1 - error\_train[r,e] / row1

accuracy\_test[r,e] = 1 - error\_test[r,e] / row2

print(w1)

print('train set accuracy: ')

print(accuracy\_train)

print('test set accuracy: ')

print(accuracy\_test)

plt.plot(accuracy\_train, lr, label='train')

plt.plot(accuracy\_test, lr, label='test')

plt.xlabel("epoch")

plt.ylabel("learning rate")

plt.legend()

plt.show()

1. **Test**
2. The result of accuracy for each epoch and each learning rate. Since there is some problem for my computer to install the tabulate library, the result will be shown in the list. The list is 3 \* 5. The three rows represent learning rate, 0.001, 0.005, 0.01 respectively. And the five columns represent five epochs respectively.

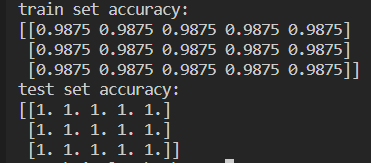


Figure 4: Accuracy

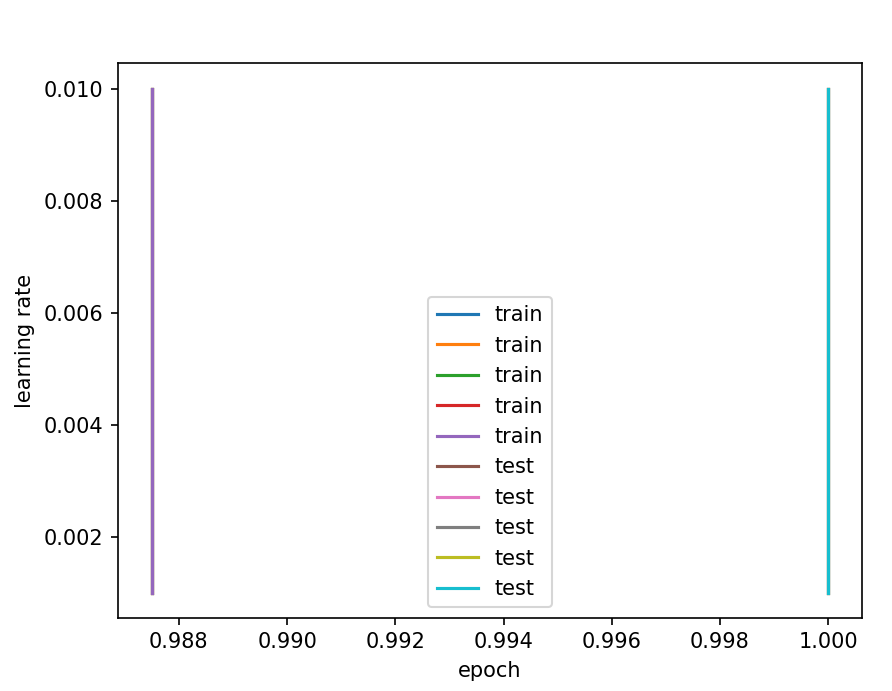
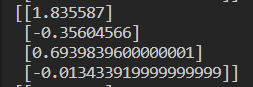


Figure 5: Result shown in the figure

*Mention: The code should be run together with functions shown in Question 1,2,3, except for function “initial” in the question 3.*

1. According to the result, it seems like the different learning rate will not affect the performance of perceptron. However, it opposites the theory. In my opinion, such problem might be caused the big gap between each learning rate, or the initial values of w and t are too close to the best one. This is because by print the values of w each epoch and each learning rate, I found they are all

.

This is quite close to the initial value,

[[1.833912],[-0.36264566],[0.70323396],[-0.01003392]], which I randomly get by the “initial” function in the Question 3.