**CPEG 586 – DEEP LEARNING**

**HOMEWORK 1**

**Name:** Ivan Sangines Escrig

**ID#:** 968606

**Instructor:** Dr. Mahmood

**Date:** February 5, 2019

**TABLE OF CONTENT**

[**INTRODUCTION**](#_Toc446970371) 3

[**SCREENSHOTS:**](#_Toc446970373) 4

[**SOURCE CODE:**](#_Toc446970374) 9

**[CONCLUSION:](#_Toc446970375)**18

**INTRODUCTION**

The porpuse of this assignment is to practice some basic python skills, working with lists, tupples, strings and reading or writing into a file.

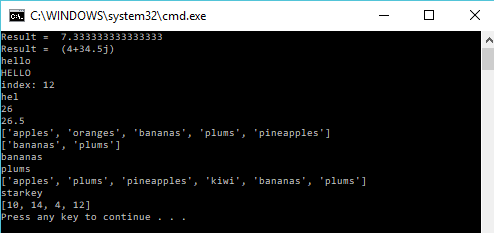
The second part of the assignment is going to help us understand better how the Linear Squares Optimization work and how to use it in order to find the better function.

The last part of the assignment will teach us how to create and train our first NN using back propagation in order to classify our test data (0 will mean the point belongs bellow the line and 1 mean the point belongs above the line).

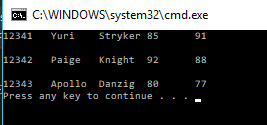
**SCREEN SHOTS:**

**Problem 1:**

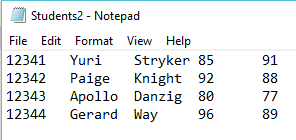
On the first Python example we did different operation with arrays and tupples in order to remember how to work with them. I used properties like remove, append and map functions.

****

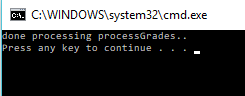
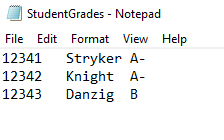
Screenshot showing the output after reading the Students.txt file:

****

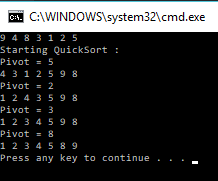
Screenshot showing the Students2.txt, which is a copy of Students.txt with Gerard added in the file.

****

The next exercise consists in reading the Students.txt file and create a Student object which will just contain id, name and grade. This new students objects will be written in a separated file.

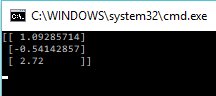
The next output will show a sorted array using QuickSort:



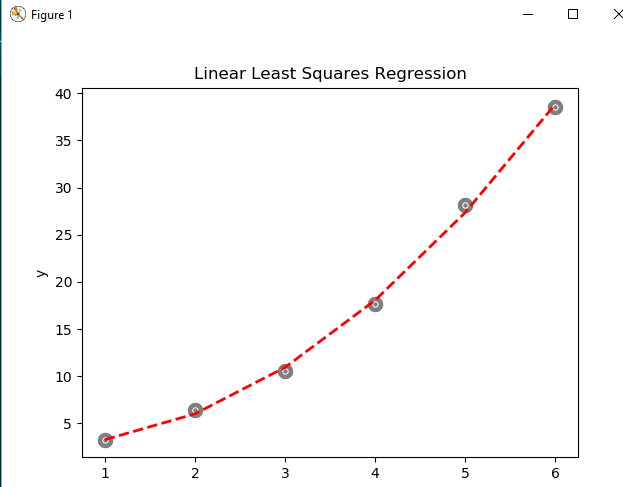
**Problem 2:**

In this exercise we are are given some data points and our goal is to find the line equation with less possible error than contains the given data points. We are solving this problem for a second order equation which has the eq. y = ax^2 +bx + C, the goal is to find a,b,C.

The first output shows the values for a,b,C.

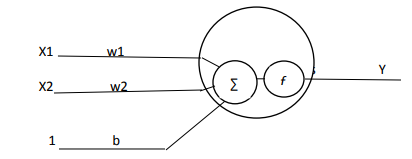
****

On the second output we can see the data points given (black dots) and the obtained line equation which is shown in red.

****

**Problem 3:**

**Part a:**



S = Summation a = actual output

S = X1 \* w1 + X2 \* w2 + 1 \* b a = X1 \* w1 + X2 \* w2 + 1 \* b

Since we are not using activation function, a = S

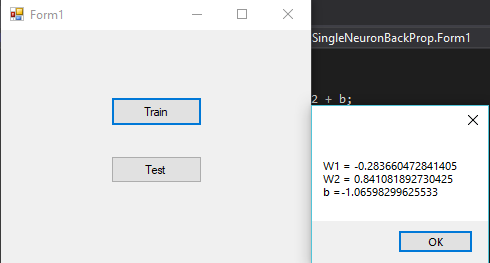
L =

Partial derivatives in order to find the gradients:

In order to obtain the adjusted gradients:

**Part b:**

In this output we can see the values for our weights after training our NN. I have added new data points since our given data was small and it was not learning properly. This weights are updated after each Epoc using the formula UpdatedWeight = OldWeight – learning rate \* gradient

****

**SOURCE CODE:**

**Problem 1:**

Arrays and Tupples:

from math import pi

import sys

def computeAvg(a,b,c) :

return (a + b + c)/3.0;

def doComplexMath() :

num1 = 3 + 4j

num2 = 6 + 3.5j

res = num1 \* num2

return res;

def mapTest(mylist) :

ys = map(lambda x: x \* 2, mylist)

#ys is a map object, so we need to convert it to a list

result = []

for elem in ys:

result.append(elem)

return result

def main():

#Simple math

print("Result = ",computeAvg(5,8,9))

print("Result = ",doComplexMath())

#working with strings

s1 = "hello"

s2 = s1.upper()

print(s1 + "\n" + s2)

s3 = "hello there how are you"

pos = s3.find('how')

print("index: " + str(pos)) #should print index 12

s4 = 'helllo'

s5 = s1[0:3]

print(s5)

snum = "25"

num1 = int(snum)

print(num1 + 1)

snum2 = "25.5"

num2 = float(snum2)

print(num2 + 1)

#list and tupples examples

fruits = ['apples', 'oranges', 'bananas', 'plums', 'pineapples']

print(fruits)

pfruits = fruits[2:4]

print(pfruits)

for fr in pfruits :

print(fr)

del fruits[2]

fruits.remove('oranges')

fruits.append('kiwi')

print(fruits + pfruits)

s1 = ('john','starkey',12341,3.5) # firstname, lastname, id, gpa

print(s1[1])

#map and lambda functions

ys = mapTest([5,7,2,6])

print(ys)

if \_\_name\_\_ == "\_\_main\_\_":

sys.exit(int(main() or 0))

File Processing:

import sys

def displayFile():

fobj = open("C:/Users/ivans\_000/Desktop/MASTER/Spring2019/Deep\_Learning/Assignment1\_Sangines/Students.txt","r")

for line in fobj:

print(line)

def copyFile():

fobj = open("C:/Users/ivans\_000/Desktop/MASTER/Spring2019/Deep\_Learning/Assignment1\_Sangines/Students.txt","r")

fobj2 = open("C:/Users/ivans\_000/Desktop/MASTER/Spring2019/Deep\_Learning/Assignment1\_Sangines/Students2.txt","w")

for line in fobj:

print(line)

fobj2.write(line)

fobj2.write("\n12344" + "\t" + "Gerard" + "\t" + "Way" + "\t" + "96" + "\t" + "89")

fobj2.close()

def main():

#displayFile()

copyFile()

if \_\_name\_\_ == "\_\_main\_\_":

sys.exit(int(main() or 0))

File Processing2:

import sys

from Student import Student

def processGrades():

fobj = open("C:/Users/ivans\_000/Desktop/MASTER/Spring2019/Deep\_Learning/Assignment1\_Sangines/Students.txt","r")

fobj2 = open("C:/Users/ivans\_000/Desktop/MASTER/Spring2019/Deep\_Learning/Assignment1\_Sangines/StudentGrades.txt","w")

for line in fobj:

parts = line.split('\t')

s1 = Student("","",0)

s1.id = parts[0]

s1.firstName = parts[1]

s1.lastName = parts[2]

s1.addTestScore(parts[3])

s1.addTestScore(parts[4])

s1.grade = s1.computeGrade()

# now write id and grade to an output file

fobj2.write(s1.id + "\t" + s1.lastName + "\t" + s1.grade + "\n")

fobj2.close()

print("done processing processGrades..")

def main():

processGrades()

if \_\_name\_\_ == "\_\_main\_\_":

sys.exit(int(main() or 0))

QuickSort:

import sys

def quicksort(arr, i, j):

if i < j:

pos = partition(arr, i, j)

quicksort(arr, i, pos - 1) # quicksort left list

quicksort(arr, pos + 1, j) # quick sort right list

def partition(arr, i, j):

pivot = arr[j]

small = i - 1

for k in range(i, j):

if arr[k] <= pivot:

small += 1

swap(arr, k, small)

printArray(arr)

swap(arr, j, small + 1)

print("Pivot = " + str(arr[small + 1]))

printArray(arr)

return small + 1

def swap(arr, i, j):

arr[i], arr[j] = arr[j], arr[i] # exchange data at index i and j

def printArray(arr):

print(' '.join(str(i) for i in arr))

#--------------------------------------------------------

def main():

arr = [9, 4, 8, 3, 1, 2, 5]

printArray(arr)

print("Starting QuickSort :")

quicksort(arr, 0, len(arr) - 1)

if \_\_name\_\_ == "\_\_main\_\_":

sys.exit(int(main() or 0))

**Problem 2:**

import sys

import numpy as np

import matplotlib.pyplot as plt

def main():

x = np.ndarray((6,1))

#x = np.zeros((6,1))

#x = x.astype(float)

y = np.ndarray((6,1))

#y = np.zeros((6,1))

#y = y.astype(float)

x[0,0] = 1

x[1,0] = 2

x[2,0] = 3

x[3,0] = 4

x[4,0] = 5

x[5,0] = 6

y[0,0] = 3.2

y[1,0] = 6.4

y[2,0] = 10.5

y[3,0] = 17.7

y[4,0] = 28.1

y[5,0] = 38.5

a=b=c=d=0

a1=b1=c1=d1=0

a2=b2=c2=d2=0

for i in range(len(x)):

a += x[i,0] \* x[i,0] \* x[i,0] \* x[i,0]

a1 += x[i,0] \* x[i,0] \* x[i,0]

a2 += x[i,0] \* x[i,0]

b += x[i,0] \* x[i,0] \* x[i,0]

b1 += x[i,0] \* x[i,0]

b2 += x[i,0]

c += x[i,0] \* x[i,0]

c1 += x[i,0]

c2 += 1

d += y[i,0] \* x[i,0] \* x[i,0]

d1 += y[i,0] \* x[i,0]

d2 += y[i,0]

A = np.ndarray((3,3))

#A = np.zeros((3,3))

#A = A.astype(float)

A[0,0] = a #int(a) #2275

A[0,1] = b #int(b) #441

A[0,2] = c #int(c) #91

A[1,0] = a1 #int(a1) #379.16666667

A[1,1] = b1 #int(b1) #73.5

A[1,2] = c1 #int(c1) #15.1666667

A[2,0] = a2 #int(a2) #63.19444

A[2,1] = b2 #int(b2) #12.25

A[2,2] = c2 #int(c2) #2.527778

ainv = np.linalg.inv(A) # Doing inverse of A

z = np.ndarray((3,1)) #Results

#z = np.zeros((3,1))

#z = z.astype(float)

z[0,0] = d

z[1,0] = d1

z[2,0] = d2

res = np.dot(ainv,z) # a = res[0,0] and b=[1,0]

print(res)

# do a scatter plot of the data

area = 10

colors = ['black']

plt.scatter(x, y, s=area, c=colors, alpha=0.5, linewidths=8) #drawing points using X,Y data arrays

plt.title('Linear Least Squares Regression')

plt.xlabel('x')

plt.ylabel('y')

yfitted = x \* x \* res[0,0] + x \* res[1,0] + res[2,0]

line, = plt.plot(x, yfitted, '--', linewidth=2) #line plot

line.set\_color('red')

plt.show()

if \_\_name\_\_ == "\_\_main\_\_":

sys. exit(int(main() or 0))

**Problem 3:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace SingleNeuronBackProp

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

double w1, w2, b;

private void btnTrain\_Click(object sender, EventArgs e)

{

w1 = 0.3;

w2 = 0.1;

b = -0.1;

double x = 0;

double[] trainingX = { 1.0, 1.0, 2.0, 2.0, 3.0, 3.0 };

double[] trainingY = { 2.2, 2.4, 2.5, 2.7, 2.8, 3.0 };

for (int j = 0; j < 1000; j++) //Epocs

{

for (int i = 0; i < trainingX.Length; i++) //Training points

{

x = i;

double newW1 = newWeightW1(trainingX[i], trainingY[i], w1, w2, b);

double newW2 = newWeightW2(trainingX[i], trainingY[i], w1, w2, b);

double newb = newBias(trainingX[i], trainingY[i], w1, w2, b);

w1 = newW1;

w2 = newW2;

b = newb;

}

}

MessageBox.Show("W1 = " + w1.ToString() + "\nW2 = " + w2.ToString() + "\nb =" + b.ToString());

}

double newWeightW1(double x1, double x2, double w1, double w2, double b)

{

// compute output

double y; //class where the point belongs 1 above 0 bellow

double a = w1 \* x1 + w2 \* x2 + b;

double line\_y = 0.3 \* x1 + 2;

if (line\_y < x2) //point bellow line

{

y = 1;

}

else //point above line

{

y = 0;

}

double gradw1 = -1 \* (y - a) \* x1;

w1 = w1 - 0.01 \* gradw1;

return w1;

}

double newWeightW2(double x1, double x2, double w1, double w2, double b)

{

double y; //class where point belongs: 1 above the line, 0 bellow

double line\_y = 0.3 \* x1 + 2;

double a = w1 \* x1 + w2 \* x2 + b;

if (line\_y < x2) //point bellow line

{

y = 1;

}

else //point above line

{

y = 0;

}

double gradw2 = -1 \* (y - a) \* x2;

w2 = w2 - 0.01 \* gradw2;

return w2;

}

double newBias(double x1, double x2, double w1, double w2, double b)

{

// compute output

double y;

double a = w1 \* x1 + w2 \* x2 + b;

double line\_y = 0.3 \* x1 + 2;

if (line\_y < x2) //point bellow

{

y = 1;

}

else

{

y = 0;

}

double gradb = -1 \* (y - a) \* 1;

b = b - 0.01 \* gradb;

return b;

}

private void btnTest\_Click(object sender, EventArgs e)

{

string output = "";

double[] TestX = { 1.5, 1.5, 2.5, 2.5 };

double[] testY = { 2.36, 2.5, 2.7, 2.8 };

for (int i=0; i < TestX.Length; i++)

{

double probability = w1 \* TestX[i] + w2 \* testY[i] + b;

if (probability < 0.5)

probability = 0;

else

probability = 1;

output += "X1= " + TestX[i] + " X2= " + testY[i] + " Output= " + probability + "\n";

}

MessageBox.Show(output);

}

}

}

**Conclusion:**

After doing this assignment, the first part related about Python examples helped me refresh some python properties such as how arrays work or how to work with files.

The second exercise helped me to understand better how Least Squares Optimization work and how it can be applied in order to find a proper equation to fit the given data with the least possible error. This can be done by doing the corresponding partial derivatives of the error function. Once the derivatives are computed, we need to obtain the summation of each variable using the different given points. After obtaining the summation of each variable, all we need to do is solve a system of equations using matrix.

The last part definitely helped me to understand the concepts explained in class about classifying data using backpropagation. I was able to understand how to train a NN using backpropagation in order to update our gradients based on the partial derivatives and the given training data.

To conclude, before starting this assignment I was confused about the differences between regression and classification, this first assignment really helped me to understand better the differences between them