Deep Learning Homework II Report

Name: Dazhi Li

RUID: 197007456

Problem 1

Source Code:

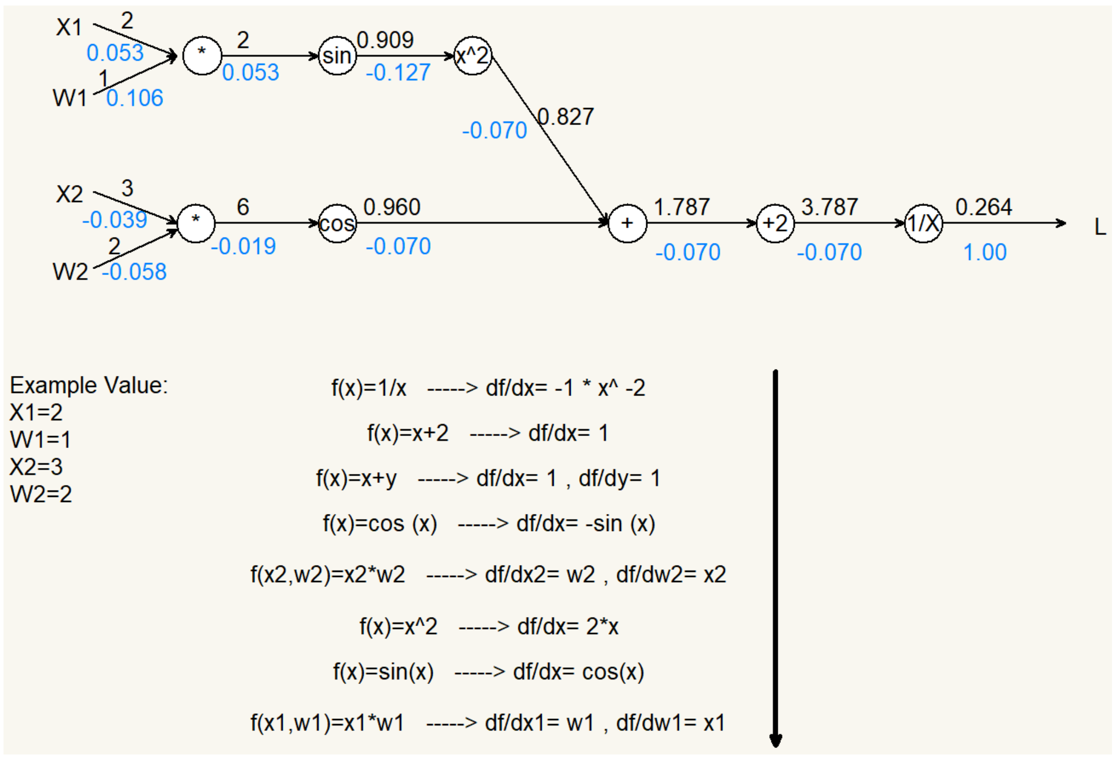
```
import math
x_1=2
w_1=1
x_2=3
w_2=2
def manual_com():
    f_1=x_1*w_1
    f_2=math.sin(f_1)
    f_3=f_2**2
    f_4=x_2*w_2
    f_5=math.cos(f_4)
    f_6=f_5+f_3
    f_7=f_6+2
    f_8=1/f_7
    d_8=1.00
    d_7=d_8*(-1/f_7**2)
    d_6=d_7*1
    d_5=d_6*1
    d_4=d_5*(-math.sin(f_4))
    d_x_2=d_4*w_2
    d_w_2=d_4*x_2
    d_3=d_5*1
    d_2=d_3*2*f_2
    d_1=d_2*math.cos(f_1)
    d_x_1=d_1*w_1
    d_w_1=d_1*x_1
    print("The result of forward propagation is "+str(f_8))
    print("The gradient of x1 is %.3f"%d_x_1)
    print("The gradient of w1 is %.3f"%d_w_1)
    print("The gradient of x2 is %.3f"%d_x_2)
    print("The gradient of w2 is %.3f"%d_w_2)


manual_com()
```

Problem procedure:

At the begging of the whole program we should set up the initial value of x1, x2, w1, w2 to compute. I draw the computational graph first to calculate the final result of forward propagation. Then I analysis all the functions used in the forward propagation and calculated out the derivate function. The principle here is, upstream gradient multiple local gradient. By this way we move forward back to the derivate respects to x1, w1, x2, w2. Those “f” function in my source code means toward propagation and I make a number to each of the computation. Those “d” function means backpropagation function.

Computational Graph:



Example Value:
X1=2
W1=1
X2=3
W2=2

f(x)=1/x -----> df/dx= -1 * x^ -2

f(x)=x+2 -----> df/dx= 1

f(x)=x+y -----> df/dx= 1 , df/dy= 1

f(x)=cos (x) -----> df/dx= -sin (x)

f(x2,w2)=x2*w2 -----> df/dx2= w2 , df/dw2= x2

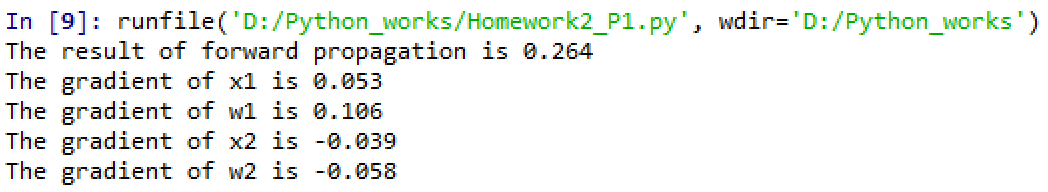
f(x)=x^2 -----> df/dx= 2*x

f(x)=sin(x) -----> df/dx= cos(x)

f(x1,w1)=x1*w1 -----> df/dx1= w1 , df/dw1= x1

Figure 1-1

Output of Problem 1:



```
In [9]: runfile('D:/Python_works/Homework2_P1.py', wdir='D:/Python_works')
The result of forward propagation is 0.264
The gradient of x1 is 0.053
The gradient of w1 is 0.106
The gradient of x2 is -0.039
The gradient of w2 is -0.058
```

Figure 1-2

Problem 2

Source Code:

```
import numpy.matlib

import numpy as np

import math

W=np.matlib

W=[[0.1,0.4,0.2],[0.2,0.5,0.7],[0.2,0.6,0.3]]

X=np.array([[0.5],[1.5],[1.1]])


def sigmoid(array):

    temp=[]
    for i in array:
        k=1/(1+math.exp(-i))
        temp.append([k])
    a=np.array(temp).reshape(3,1)
    return  a

def L2(array):
    sum=0
    for i in array:
        sum+=i**2
    return sum

def forward_pass():
    global f_1
    f_1 = numpy.dot(W, X)
    global f_2
    f_2 = sigmoid(f_1)
    global f_3
    f_3 = L2(f_2)
    print("W * X result:\n",f_1)
    print("\nSigmoid result:\n",f_2)
    print("\nL2 distance result:\n",f_3)

def W_derivate_cal():
    temp=[]
    for i in np.nditer(d_1):
        for j in np.nditer(X):
            Mij=i*j
            temp.append(Mij)
    a=np.array(temp).reshape(3,3)
    return a

def X_derivate_cal():
    temp=[]
    for i in range(3):
        xi=0
        for j in range(3):
            xi+=d_1[j][0]*W[j][i]
        temp.append(xi)
    a=np.array(temp).reshape(3,1)
    return a

def back_propagation():
    global d_3
    d_3=1
    global d_2
    d_2=d_3 * 2 * f_2
    global d_1
    d_1=d_2*(1-sigmoid(f_1))*sigmoid(f_1)
    d_w=W_derivate_cal()
    d_x=X_derivate_cal()
    print("\nThe derivate of L2 is:\n",d_2)
    print("\nThe derivate of Sigmoid is:\n",d_1)
    print("\nThe derivate respects to w is:\n",d_w)
    print("\nThe derivate respects to x is:\n",d_x)


forward_pass()
back_propagation()
```

Program Procedure:

This program is doing forward propagation computation first. I defined a few key functions like sigmoid and L2 distance calculation function. And those functions are designed for matrix calculation. Matrix is kind of like ndarray, but only two dimensions in a matrix. So we can still use array as input but I chose matlib to describe matrix W. The hardest point is matrix calculation which could be done by the numpy.dot() function. Then what I am going to solve is how to achieve back propagation through the whole procedure. Returning from the L2 distance function and sigmoid is not so hard as we can calculated the derivate function by myself. However, finding derivate respects to W and X is not so easy. I firstly calculated how Pi is generated by the multiplication of W and X matrix. Then I am returning back to find the W and X where Wij=qi * xj and xi=Σ(qj*wij) as their local gradient. Finally, I run my back propagation function to find derivate respects to W and X.

Computational Graph:

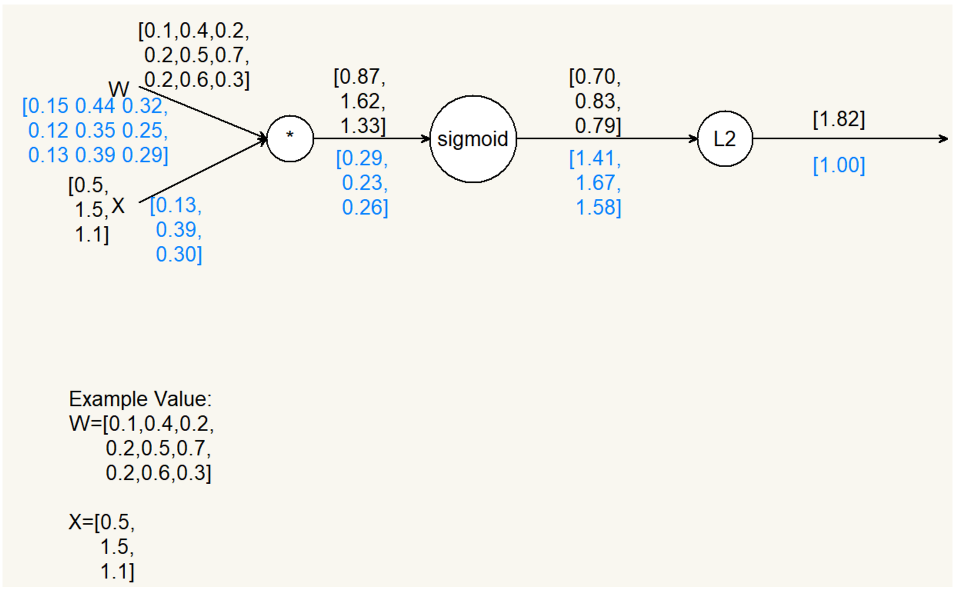
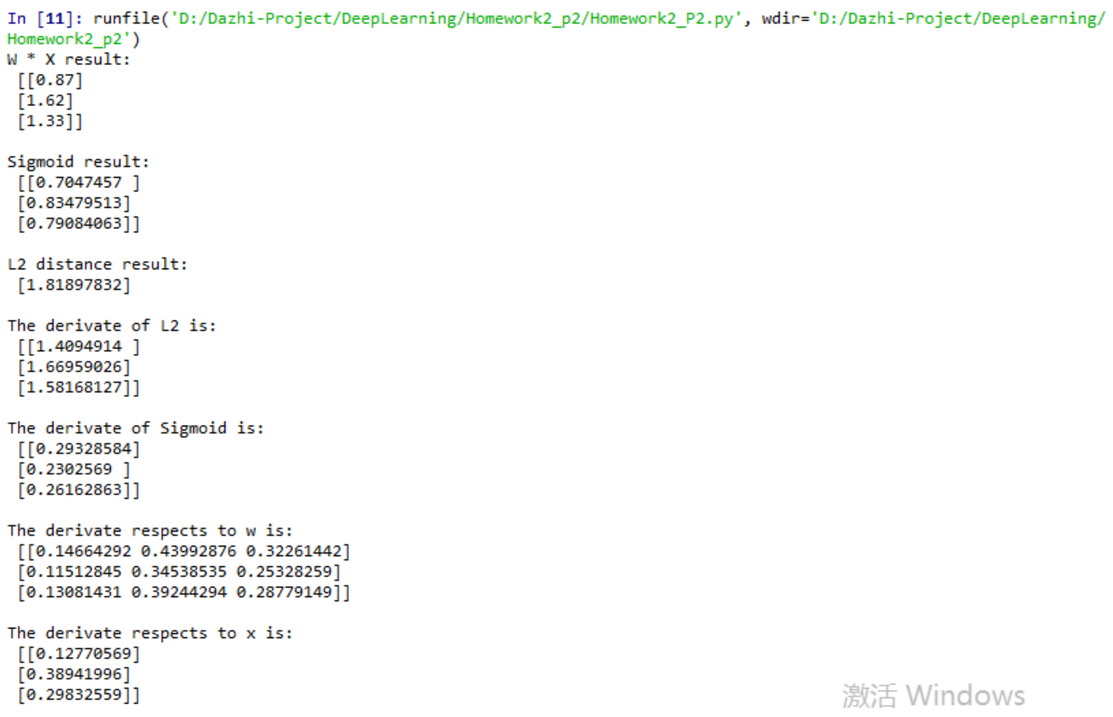


Figure 2-1

Output of Problem 2:



```
In [11]: runfile('D:/Dazhi-Project/DeepLearning/Homework2_p2/Homework2_P2.py', wdir='D:/Dazhi-Project/DeepLearning/
Homework2_p2')
W * X result:
 [[0.87]
 [1.62]
 [1.33]]

Sigmoid result:
 [[0.7047457 ]
 [0.83479513]
 [0.79084063]]

L2 distance result:
 [1.81897832]

The derivate of L2 is:
 [[1.4094914 ]
 [1.66959026]
 [1.58168127]]

The derivate of Sigmoid is:
 [[0.29328584]
 [0.2302569 ]
 [0.26162863]]

The derivate respects to w is:
 [[0.14664292 0.43992876 0.32261442]
 [0.11512845 0.34538535 0.25328259]
 [0.13081431 0.39244294 0.28779149]]

The derivate respects to x is:
 [[0.12770569]
 [0.38941996]
 [0.29832559]]
```

Figure 2-2