Himani Parikh ID#:- 1322085

Homework 1 for DTSC740 (Deep Learning)
Due October 21, 2023

Attention: Please submit your own work - Copying will be penalized.

1. (25 points: Practice of scalar-based backpropagation)You are required to calculate the gradients of

$$f(\boldsymbol{x}, \boldsymbol{w}) = \frac{1}{2 + sin^2(x_1 w_1) + cos(x_2 w_2)}$$

   with respect to w_1, w_2.
   (a) Use computational graph for calculation.
   (b) Based on (a), writhe a program to implement the computational graph and verify your answer in (a).

   Note: You are free to choose the inputs of your computational graph (values for x_1, x_2, w_1, w_2)

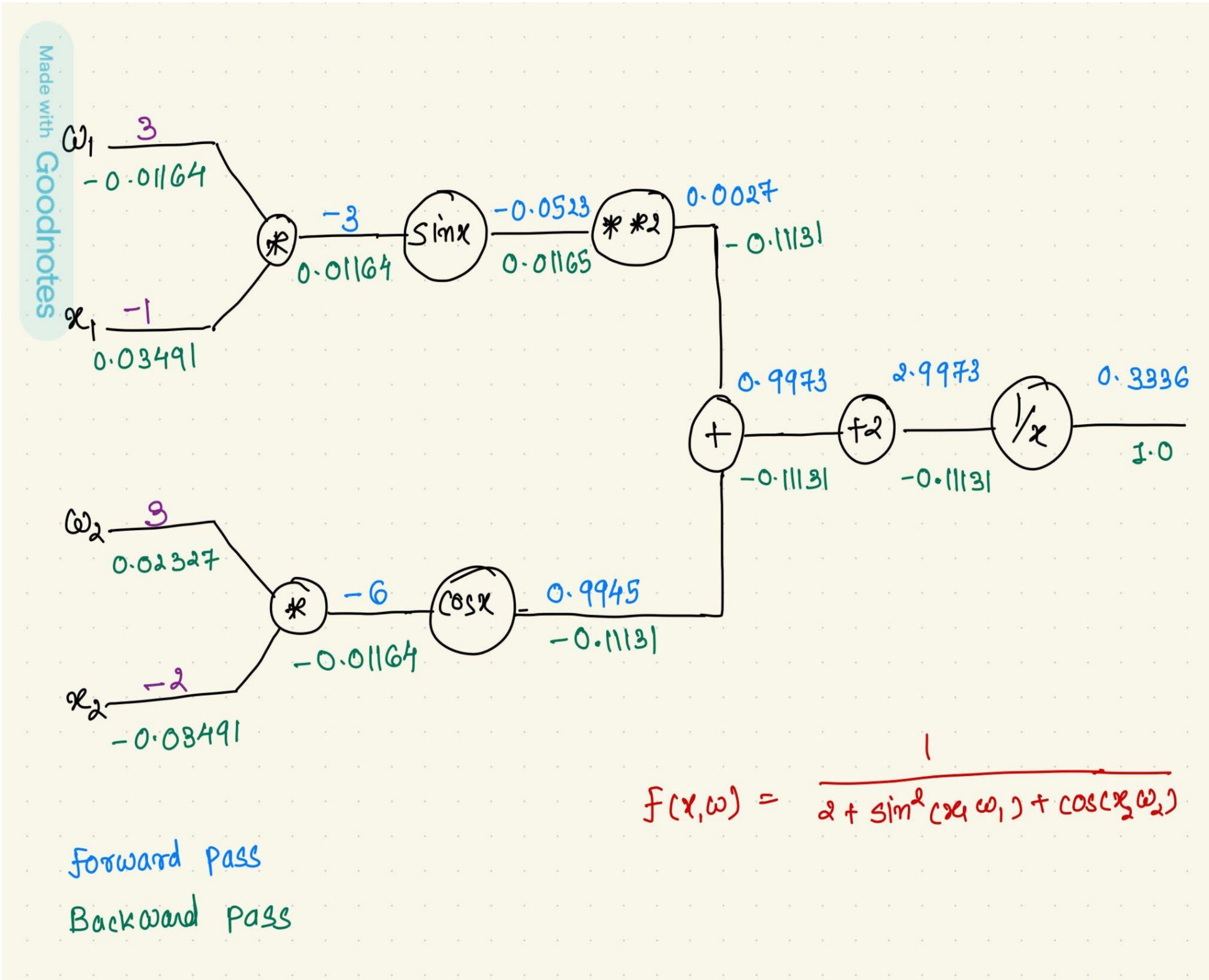
2. (25 points: Practice of vector-based backpropagation). You are required to calculate the gradients of

$$f(\boldsymbol{x}, \boldsymbol{w}) = ||\sigma(\boldsymbol{W}\boldsymbol{x})||^2$$

   with respect to $W_{i,j}$.
   Here $|| \cdot ||^2$ is the calculation of L_2 loss function, $\boldsymbol{W}$ is 3-by-3 matrix and $\boldsymbol{x}$ is 3-by-1 vector, and $\sigma(\cdot)$ is the sigmoid function that performs *element-wise* sigmoid operation.
   (a). Use computational graph for calculation
   (b). Based on(a), write a program to implement the computational graph and verify your answer in (a).
   (c). (Optional - extra 2 points) Use the vectorized approach in python to simply your codes.

   What to submit: a PDF file for this problem, including source codes and print screen results.

Question1



Code:

```
import math
import sympy as sp

# Define the input values
x1 = -1
w1 = 3
x2 = -2
w2 = 3

# Define the forward pass for each gate
```

```
print("***********************Output for each gate in Forward Pass:************************")
print()

print()

print("--------------------------Forward pass (x1*w1)----------------------------")
print()
mul_x1w1 = x1*w1
print("Output of multiplication:", mul_x1w1)
print()

print("--------------------------Forward pass (Sinx)------------------------------")
print()
sin_value = math.sin(math.radians(mul_x1w1))
print("Output of sin:", round(sin_value, 4))
print()

print("--------------------------Forward pass (**2)------------------------------")
print()
sin_squared = math.sin(sin_value) ** 2
print("Output of sin_squared:", round(sin_squared, 4))
print()

print("--------------------------Forward pass (x2*w2)-----------------------------")
print()
mul_x2w2 = x2 * w2
print("Output of multiplication:", mul_x2w2)
print()

print("--------------------------Forward pass (cosx)------------------------------")
print()
cos_value = math.cos(math.radians(mul_x2w2))
print("Output of cos:", round(cos_value, 4))
print()

print("--------------------------Forward pass (Sin**2x + cosx)--------------------")
print()
add = sin_squared + cos_value
print("Output of addition:", round(add, 4))
print()

print("--------------------------Forward pass (+2)------------------------------")
print()
add1 = add + 2
```

```
print("Output of addition with two:", round(add1, 4))
print()

print("-------------------------Forward pass (1/x)*------------------------------")
print()
final  = 1 / add1
print("Output of 1/x:", round(final, 4))

# Define the Backward pass for each gate

print("***********************Output for each gate in Backward Pass:************************")
print()

upstream = 1.0;
x = sp.symbols('x')
w = sp.symbols('w')

print()

print("-------------------------Backpropogation (1/x)------------------------------")
print()
#local
df_dx_1 = sp.diff(1/x, x)
print("Partial derivative of 1/x with respect to x:", df_dx_1)
# current = upstream * local
current_1 = upstream * df_dx_1.subs(x, add1)
print(round(current_1, 5))

print()

print("-------------------------Backpropogation (+2)------------------------------")
print()
df_dx_2 = sp.diff(2 + x, x)
print("Partial derivative of 2 with respect to x:", df_dx_2)
current_2 = current_1 * df_dx_2
print(round(current_2, 5))
print()

print("-------------------------Backpropogation (+)------------------------------")
print()
df_dx_3 = sp.diff(x, x)
print("Partial derivative of x with respect to x for x1, w1:", df_dx_3)
current_3 = current_2 * df_dx_3
print(round(current_3, 5))
```

```
print()

print("-------------------------Backpropogation (**2)------------------------------")
print()
df_dx_4 = sp.diff(pow(x,2), x)
print("Partial derivative of **2 with respect to x:", df_dx_4)
current_4 = current_3 * df_dx_4.subs(x, sin_value)
print(round(current_4, 5))
print()

print("-------------------------Backpropogation (sinx)------------------------------")
print()
df_dx_5 = sp.diff(sp.sin(x), x)
print("Partial derivative of sinx with respect to x:", df_dx_5)
current_5 = current_4 * df_dx_5.subs(x, math.radians(mul_x1w1))
print(round(current_5, 5))
print()

print("-------------------------Backpropogation (w1*x1 wrt w1)------------------------------")
print()
df_dx_6 = sp.diff(x*w, w)
print("Partial derivative of w1*x1 with respect to w1:", df_dx_6)
current_6 = current_5 * df_dx_6.subs(x, x1)
print(round(current_6, 5))
print()

print("-------------------------Backpropogation (w1*x1 wrt x1)------------------------------")
print()
df_dx_7 = sp.diff(x*w, x)
print("Partial derivative of w1*x1 with respect to x1:", df_dx_7)
current_7 = current_5 * df_dx_7.subs(w, w1)
print(round(current_7, 5))
print()

print("-------------------------Backpropogation (+)------------------------------")
print()
df_dx_8 = sp.diff(x, x)
print("Partial derivative of x with respect to x for x2, w2:", df_dx_8)
current_8 = current_2 * df_dx_8
print(round(current_8, 5))
print()

print("-------------------------Backpropogation (cosx)------------------------------")
print()
```

```
df_dx_9 = sp.diff(sp.cos(x), x)
print("Partial derivative of cosx with respect to x:", df_dx_9)
current_9 = current_8 * df_dx_9.subs(x, math.radians(mul_x2w2))
print(round(current_9, 5))
print()

print("--------------------------Backpropogation (w2*x2 wrt w2)-------------------------------")
print()
df_dx_10 = sp.diff(x*w, w)
print("Partial derivative of w2*x2 with respect to w2:", df_dx_10)
current_10 = current_9 * df_dx_10.subs(x, x2)
print(round(current_10, 5))
print()

print("--------------------------Backpropogation (w2*x2 wrt X2)-------------------------------")
print()
df_dx_11 = sp.diff(x*w, x)
print("Partial derivative of w2*x2 with respect to x2:", df_dx_11)
current_11 = current_9 * df_dx_11.subs(w, w2)
print(round(current_11, 5))
```

Output:

```
***********************Output for each gate in Forward Pass:************************

--------------------------Forward pass (x1*w1)----------------------------

Output of multiplication: -3

--------------------------Forward pass (Sinx)------------------------------

Output of sin: -0.0523

--------------------------Forward pass (**2)------------------------------

Output of sin_squared: 0.0027

--------------------------Forward pass (x2*w2)-----------------------------

Output of multiplication: -6

--------------------------Forward pass (cosx)------------------------------

Output of cos: 0.9945

--------------------------Forward pass (Sin**2x + cosx)--------------------

Output of addition: 0.9973

--------------------------Forward pass (+2)------------------------------

Output of addition with two: 2.9973

--------------------------Forward pass (1/x)*------------------------------

Output of 1/x: 0.3336
```

```
***********************Output for each gate in Backward Pass:************************


--------------------------Backpropogation (1/x)--------------------------------

Partial derivative of 1/x with respect to x: -1/x**2
-0.11131

--------------------------Backpropogation (+2)--------------------------------

Partial derivative of 2 with respect to x: 1
-0.11131

--------------------------Backpropogation (+)--------------------------------

Partial derivative of x with respect to x for x1, w1: 1
-0.11131

--------------------------Backpropogation (**2)--------------------------------

Partial derivative of **2 with respect to x: 2*x
0.01165

--------------------------Backpropogation (sinx)--------------------------------

Partial derivative of sinx with respect to x: cos(x)
0.01164

--------------------------Backpropogation (w1*x1 wrt w1)--------------------------------

Partial derivative of w1*x1 with respect to w1: x
-0.01164

--------------------------Backpropogation (w1*x1 wrt x1)--------------------------------

Partial derivative of w1*x1 with respect to x1: w
0.03491

--------------------------Backpropogation (+)--------------------------------

Partial derivative of x with respect to x for x2, w2: 1
-0.11131

--------------------------Backpropogation (cosx)--------------------------------

Partial derivative of cosx with respect to x: -sin(x)
-0.01164
```

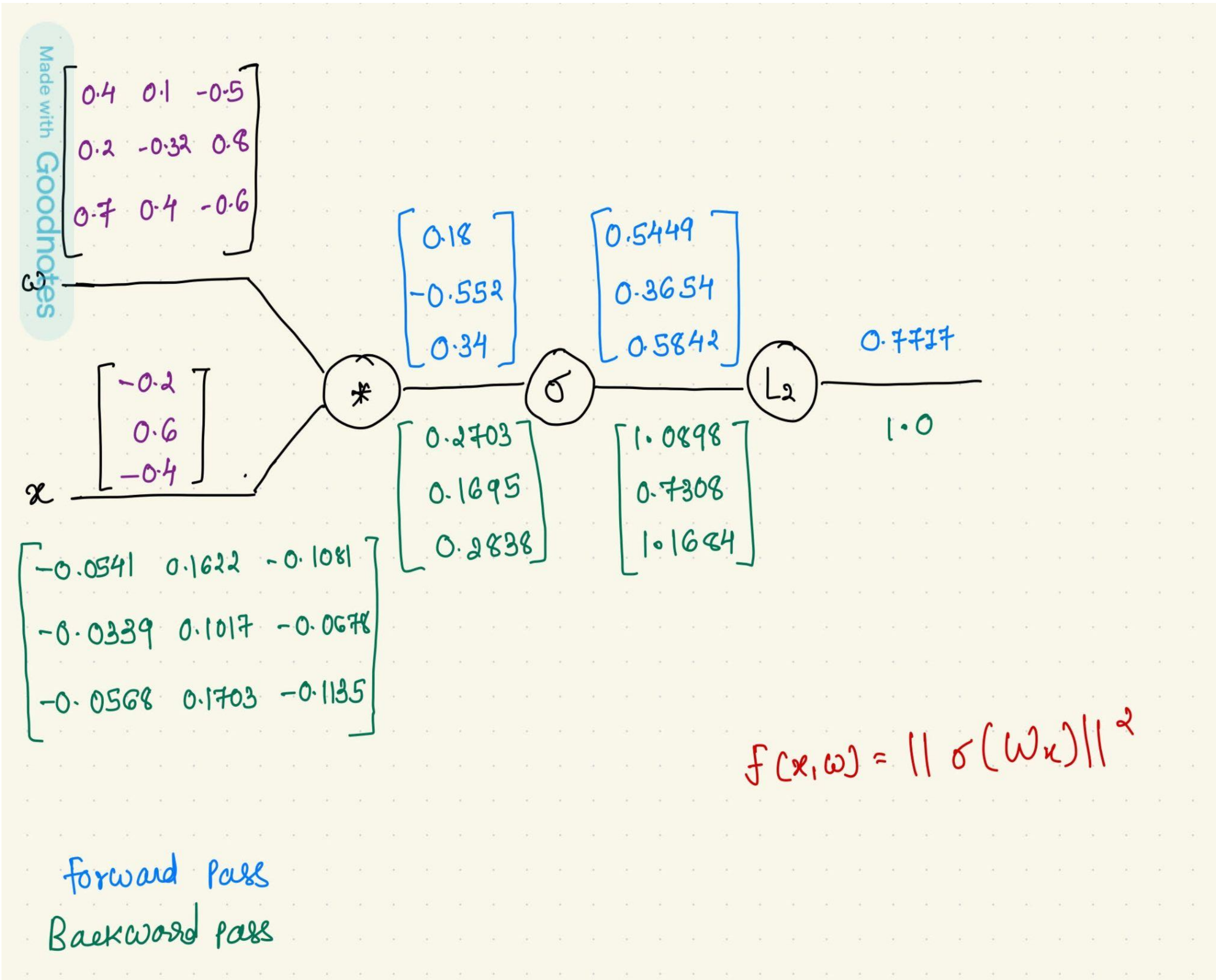
```
--------------------------Backpropogation (w2*x2 wrt w2)--------------------------------

Partial derivative of w2*x2 with respect to w2: x
0.02327

--------------------------Backpropogation (w2*x2 wrt X2)--------------------------------

Partial derivative of w2*x2 with respect to x2: w
-0.03491
```

Question : 2



Code:

```
import math
import sympy as sp
import numpy as np

# Define the input values
vector_x = np.array([[-0.2], [0.6], [-0.4]])


matrix_w = np.array([
                [0.4, 0.1, -0.5],
                [0.2, -0.32, 0.8],
                [0.7, 0.4, -0.6]
```

```
                    ])

# Define Function

def f_round_vector(arr, decimal_places):
  return np.round(arr, decimal_places)

def f_round_list(values, decimal_places):
  round_values = [round(value, 4) for value in values]
  return round_values

def f_round_matrix(arr, decimal_places):
  rounded_matrix = [[round(val, decimal_places) for val in row] for row in arr]
  return rounded_matrix

def f_matrix_multiplication(matrix, vector):
  return np.dot(matrix, vector)

def f_sigmoid(vector):
  return 1 / (1 + np.exp(-vector))

def f_L2_norm(vector):
  return np.linalg.norm(vector)

# Define the forward pass for each gate

print("************************Output for each gate in Forward Pass:************************")
print()

print("-------------------------Forward pass Multiplication-------------------------------")
print()

matrix_mul = f_matrix_multiplication(matrix_w, vector_x)
print("Output of matrix_multiplication of W and X:", matrix_mul)
print()

print("-------------------------Forward pass Sigmoid-----------------------------")
print()
sigmoid_values = f_sigmoid(matrix_mul)
sigmoid_values = f_round_vector(sigmoid_values, 4)
print("Output of sigmoid function:", sigmoid_values)
print()

print("-------------------------Forward pass L2 Norm------------------------------")
```

```
print()

norm = f_L2_norm(sigmoid_values)
norm = f_round_vector(norm, 4)
print("Output of L2 norm of the vector:", pow(norm,2))

# Define the Backward pass for each gate

upstream = 1;
a = sp.symbols('a')
x1, x2, x3 = sp.symbols('x1 x2 x3')
# vector = [x1, x2, x3]
x = sp.symbols('x')

# Define the elements of the 3x1 matrix
x1 = sp.Symbol('x1')
x2 = sp.Symbol('x2')
x3 = sp.Symbol('x3')

# Create the 3x1 matrix
# vector_3x1 = sp.Matrix([x1]; [x2]; [x3])

x_3x1 = sp.Matrix([[x1], [x2], [x3]])

print(x_3x1)
print("***********************Output for each gate in Backward Pass:*************************")
print()

print("-------------------------Backpropogation L2 Norm------------------------------")
print()

l2_norm = (x**2)
partial_derivatives = sp.diff(l2_norm, x)

print("Partial Derivatives equation of L2 norm with respect to x:", partial_derivatives)
# Output of sigmoid function: [0.5449 0.3654 0.5842]

B_l1=partial_derivatives.subs(x, sigmoid_values[0,0])
B_l2=partial_derivatives.subs(x, sigmoid_values[1,0])
B_l3=partial_derivatives.subs(x, sigmoid_values[2,0])

Back_L2 = np.array([[B_l1], [B_l2], [B_l3]])

l2_back = np.multiply(upstream , Back_L2)
```

```
print(l2_back)

print()
print("--------------------------Backpropogation Sigmoid------------------------------")
print()

sigmoid_derivative = sp.diff(1 / (1 + sp.exp(-x)), x)
print("Partial Derivatives equation of the sigmoid function wrt x:", sigmoid_derivative)
pd_sigmoid1 = sigmoid_derivative.subs(x, matrix_mul[0,0])
pd_sigmoid2 = sigmoid_derivative.subs(x, matrix_mul[1,0])
pd_sigmoid3 = sigmoid_derivative.subs(x, matrix_mul[2,0])
Back_sig = np.array([[pd_sigmoid1],[pd_sigmoid2],[pd_sigmoid3]])
sigmoid_back = np.multiply(l2_back, Back_sig)
# sigmoid_back = f_round_list(sigmoid_back, 4)
print(sigmoid_back)

print()
print("------------------------Backpropogation Multiplication--------------------------")
print()



print()
print("Partial Derivatives equation of the multiplication wrt wij: sigmoid_back * xj")


multi_back = np.outer(sigmoid_back, vector_x)
multi_back = f_round_matrix(multi_back, 4)
for row in multi_back:
    print(row)
```

Output:

```
***********************Output for each gate in Forward Pass:************************

--------------------------Forward pass Multiplication------------------------------

Output of matrix_multiplication of W and X: [[ 0.18 ]
 [-0.552]
 [ 0.34 ]]

--------------------------Forward pass Sigmoid------------------------------

Output of sigmoid function: [[0.5449]
 [0.3654]
 [0.5842]]

--------------------------Forward pass L2 Norm------------------------------

Output of L2 norm of the vector: 0.7717622499999999
```

```
Matrix([[x1], [x2], [x3]])
***********************Output for each gate in Backward Pass:************************

--------------------------Backpropogation L2 Norm------------------------------

Partial Derivatives equation of L2 norm with respect to x: 2*x
[[1.08980000000000]
 [0.730800000000000]
 [1.16840000000000]]

--------------------------Backpropogation Sigmoid------------------------------

Partial Derivatives equation of the sigmoid function wrt x: exp(-x)/(1 + exp(-x))**2
[[0.270255017494056]
 [0.169460081438825]
 [0.283818329212643]]

------------------------Backpropogation Multiplication--------------------------


Partial Derivatives equation of the multiplication wrt wij: sigmoid_back * xj
[-0.0541, 0.1622, -0.1081]
[-0.0339, 0.1017, -0.0678]
[-0.0568, 0.1703, -0.1135]
```