Comp 642: Assignment #2 (80 points)

Rice University — Due Date: Thursday, 03/09/2023

**Submission Instructions:** For coding questions, please submit python notebook along with all the plots

and 2-3 paragraphs explaining what you observe and what are your conclusions. Please do familiarize

yourself with python notebooks (like Jupyter or Google Collab), they are very convenient, you can run

them in your browser without any installations.

**1 Forward and Back-propagation in Neural Network - 20 Points**

Answer:

STEP1: we create a numpy matrix of 3\*2 for X with the values of X as given [[1,-2], [5,-9],[-3,2]] and Y matrix of (3\*) with values [[01], [1], [0.8]]

STEP2 : we assign W1 initial values to 6 variables and use these variables to build a numpy array.

W1(11) = 1 , W1(12) =0, W1(13) =1, W1(21) = -1 ,W1(22) = 0 , W1(23) =-1

STEP3: we create a numpy matrix of (2\*3) [[W1(11), W1(12), W1(13)], [W1(21), W1(22), W1(23)]]

STEP4 : Similarly create a W2 matrix with elements

[W2(1), W2(2), W2(3)] WHERE W2(1) = 1, W2(2) = 0 AND W2(3) =-1

STEP5: Forward propagation:

Step A) Apply matrix multiplication on X \* W1 which will result in a Matrix temp1 of size (3\*3)

Step B) Vectorize RELU function and apply over the whole matrix to result in a temp2 (3\*3) matrix

Step C) Apply matrix multiplication of temp2 \* W2(transpose) which will yield a matrix of (3\*1) temp3

Step D) vectorize sigmoid function and apply it over temp3 matrix which will yield yhat matrix of (3\*1)

STEP6: Calculate LOSS function:

Step A) apply matrix subtraction Y- yhat – call it temp4

Step B) Apply numpy.square function and apply 1/n\*numpy.sum on temp4 to yield Loss (L)

Note: Numpy.sum and numpy.square are vectorized operations so they are very efficient

STEP7 : Calculating the gradiants:

Step A) Apply vectorized functions (LOSS, [W2(1), W2(2), W2(3)]) which will yield a matrix of

W2Grad1 = [, , , ]

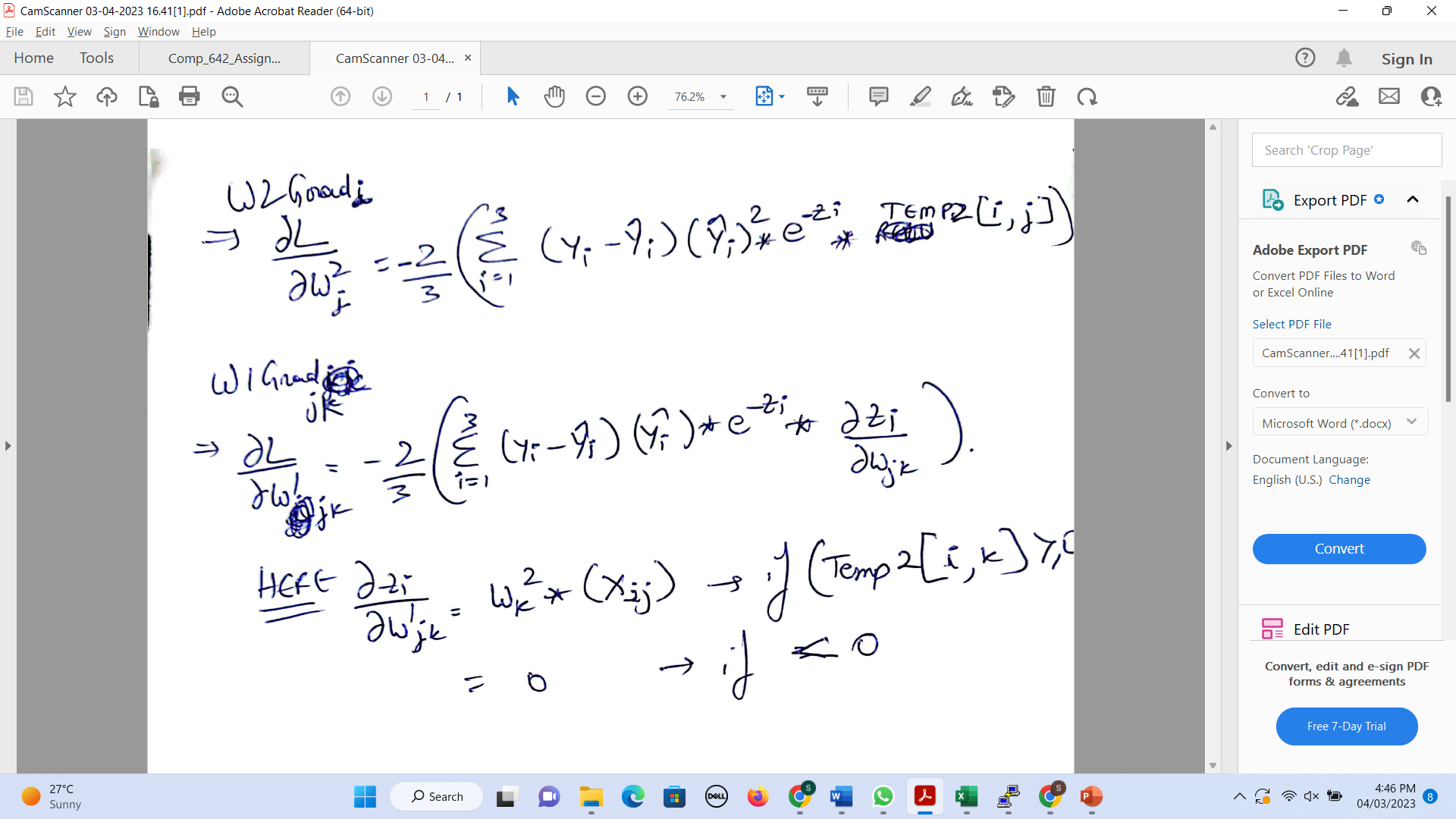
Here W2Gradi is as given below

Step B) Apply vectorized functions (LOSS, [W1(11), W1(12), W1(13), W1(21), W1(22), W1(23)]) which will yield a matrix of Grad2 =[ [, , ,][, , ]]

Here W1Gradi,j is given as below

Step C) Now calculate new W2 matrix as W2 – α \*Grad1

Step D) Now calculate new W1 matrix as W1 – α \*np.reshape(Grad1,(3,2))



STEP 8: we jump back to Step2 with new W1 and W2 Matrixes and calculate new LOSS values and keep a track of the loss values. Will continue the loop till the time the LOSS value stops reducing and those will be the final W1 and W2 matrixes

Calculations in following pages:

