CSE-5368-001Neural Networks
Fall 2021 Exam 1

Last Name	First Name	ID

Prob #	1	2	3	4	5
Points	10	20	20	25	25

Time: 80 Minutes

NOTES:

a. **Credit is only given to the correct numerical values.**
b. **All numerical values must be calculated with three digits of accuracy after the decimal point.**

1. Consider a multi-layer neural network with two nodes at the last layer. The output of this network for a given input is shown below:

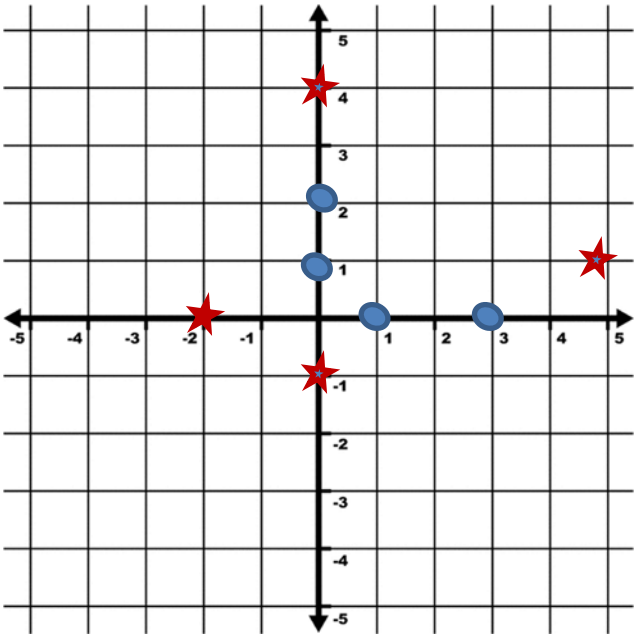
$$\textbf{actual output}=\begin{bmatrix}\mathbf{1}\\\mathbf{2}\end{bmatrix}$$

The desired output for this input is:

$$\textbf{desired output}=\begin{bmatrix}\mathbf{2}\\\mathbf{-1}\end{bmatrix}$$

Using softmax, calculate the cross entropy loss. Use natural log.

2. Design a two-layer Perceptron neural network which will correctly classify the two classes (circles and stars) as shown below. Assume the activation (transfer) function for all the nodes are hard-limit with the output of **0 (star)** and **1(Circle)**.



Show the weight matrices and biases for both layers. Biases should be included in the weight matrices in the first column.

Weight matrix for the first layer =

Weight matrix for the second layer =

3. Consider the expression:

$$\boldsymbol{f(x,y) = \frac{200}{xy} + [\max(x, xy)]^2}$$

Draw the computational graph for this expression and compute the numerical values of the partial derivatives with respect to x and y given the inputs:

$$\boldsymbol{x = 5,\ \ y = 2}$$

$\boldsymbol{\frac{\delta f(x,y)}{\delta x} =}$
$\boldsymbol{\frac{\delta f(x,y)}{\delta y} =}$

4. Given a single-layer linear associator neural network. Complete the code for the following function (this is similar to the function in the assignment 02).

- **Only implement the delta learning $W_{new} = W_{old} + \alpha(t - a)p^T$**
- **Ignore bias**

```
import numpy as np
```

```
def train(X, Y, W, batch_size, num_epochs, alpha):
""" Given a set of data points, and the hyperparameters, this function adjusts
the weights using the delta learning rule.
:param X: Array of input [input_dimensions,n_samples]
:param y: Array of desired (target) outputs [number_of_nodes,n_samples].
:param W: Array of weights (number_of_nodes,input_dimensiona)
:param num_epochs: No. of times training should be repeated over all data
:param batch_size: number of samples in a batch
:param alpha: Learning rate
:return Adjusted weights:"""
```

5. Complete the following code using Tensorflow. The code should initialize the weights and biases and compute the output, loss, and gradients, and perform a single weight update of a neural network. Assume the dimension of each input sample is 400. There are 100 nodes with sigmoid activation in the first layer, and 10 output nodes with linear activation. Loss is MSE. Assume you have a batch of inputs called X that has dimensions [20, 400], and a batch of targets called y that has dimensions [20, 10]

```
import tensorflow as tf
import numpy as np
# Create random weights and biases
W_1 = tf.Variable(np.random.randn(                ), trainable=True)
b_1 = tf.Variable(np.random.randn(                ), trainable=True)
W_2 = tf.Variable(np.random.randn(                ), trainable=True)
b_2 = tf.Variable(np.random.randn(                ), trainable=True)
with tf.GradientTape(persistent=True) as tape:
# Calculate output


output=
# Calculate loss


Loss=
# Calculate gradients

dW_1, db_1 =

dW_2, db_2 =
# Calculate new values of weights and biases
```