CSE-5368 Neural Networks Fall 2023 Exam 1_

Last Name	First Name	Student ID Number

Prob #	**1**	**2**	**3**	**4**	**Total**
Points	**25**	**25**	**25**	**25**	

Time: 80 Minutes

Last Name	First Name	Student ID Number

$$F(\mathbf{x}) = F(\mathbf{x}^*) + \nabla F(\mathbf{x})^T\Big|_{\mathbf{x}=\mathbf{x}^*}(\mathbf{x}-\mathbf{x}^*)$$

$$+\frac{1}{2}(\mathbf{x}-\mathbf{x}^*)^T\nabla^2 F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*}(\mathbf{x}-\mathbf{x}^*) + \cdots$$

$$\frac{\mathbf{p}^T\nabla F(\mathbf{x})}{\|\mathbf{p}\|} \qquad \frac{\mathbf{p}^T\nabla^2 F(\mathbf{x})\mathbf{p}}{\|\mathbf{p}\|^2} \qquad \alpha_k = -\frac{\mathbf{g}_k^T\mathbf{p}_k}{\mathbf{p}_k^T\mathbf{A}\mathbf{p}_k}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k\mathbf{g}_k \qquad \mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{p}_k$$

$$L_i = \sum_{j\neq i} max(0, y_j - y_i + \Delta)$$

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

$$H(p,q) = -\sum_x p(x)log(q(x))$$

$$L_i = -log\left(\frac{e^{y_i}}{\sum_j e^{y_j}}\right)$$

Last Name	First Name	Student ID Number

1. Consider the following performance surface

$$\boldsymbol{F(X) = 2x_1^2 + x_2^2 - 2x_1x_2 + 5}$$

Given the initial point $\begin{bmatrix} -1.5 \\ -2 \end{bmatrix}$, take **two steps** of the **steepest descent algorithm**, minimizing along a line **at each step**.

You must show your steps and the resulting positions after each step.

Last Name	First Name	Student ID Number

Problem 1 Continued

Last Name	First Name	Student ID Number

2. Complete the code for the following function.

This function is the same as the **initialization part of the Assignment_01**.

```
import numpy as np

def multi_layer_nn(X_train,Y_train,layers):
# This function Initializes all the weights and returns weight
# matrices in a list
# bias should be included in the weight matrix.
# X_train: Array of input for training [input_dimension,nof_train_samples]
# Y_train: Array of desired outputs [output_dimension,nof_train_samples]
# layers: array of integers representing number of nodes in each layer
# return: This function should return a list of weight matrices.
# Each element of the list should be a numpy array corresponds to the
# weight matrix of the corresponding layer.
# Initialize all the weights to zero
```

___

___

___

___

___

___

___

___

___

___

___

___

___

___

___

___

___

Last Name	First Name	Student ID Number

Problem 2 Continued

Last Name	First Name	Student ID Number

3. Using tensorflow, complete the following function to create and train a two-layer neural network. The **first layer** has **7 sigmoid** nodes. The **output layer** has **linear nodes**. Loss function should be **MSE**. Anything not specified in the description should be **inferred from the function's arguments and not hardcoded.**
   Code should include initializing weights, training loop with forward pass, gradient calculation, and weight updates.
   You may assume the entire dataset is one batch.

```
import numpy as np
import tensorflow as tf
def create_and_train_nn(X, Y, epochs, alpha):
"""
:param X: Array of input [n_samples,input_dimensions]
:param y: Array of desired outputs [n_samples , target_dimension].
:param epochs: number of epochs
:param alpha: Learning rate:
:return w1, w2 (w1 and w2 are the weight matrices after the training is done)."""
```

Last Name	First Name	Student ID Number

Problem 3 Continued

Last Name	First Name	Student ID Number

4. Consider the following training set for a Perceptron neural network with hard-limit activation functions.

$$\left\{p_1 = \begin{bmatrix}0\\0\end{bmatrix}, t_1 = \begin{bmatrix}0\\1\end{bmatrix}\right\}, \left\{p_2 = \begin{bmatrix}1\\2\end{bmatrix}, t_2 = \begin{bmatrix}0\\1\end{bmatrix}\right\}, \left\{p_3 = \begin{bmatrix}0\\-2\end{bmatrix}, t_3 = \begin{bmatrix}1\\0\end{bmatrix}\right\},$$

$$\left\{p_4 = \begin{bmatrix}3\\0\end{bmatrix}, t_4 = \begin{bmatrix}1\\0\end{bmatrix}\right\}, \left\{p_5 = \begin{bmatrix}1\\0\end{bmatrix}, t_5 = \begin{bmatrix}1\\1\end{bmatrix}\right\}, \left\{p_6 = \begin{bmatrix}3\\1\end{bmatrix}, t_6 = \begin{bmatrix}1\\1\end{bmatrix}\right\}$$

Design a Perceptron network with **one layer** and **two nodes** to solve this problem. Show the weight matrix. Biases should be included in the weight matrix in the first column.

Last Name	First Name	Student ID Number

Problem 4 Continued