**Department of Computing**

**CS370: Artificial Intelligence**

**Class: BSCS-10AB**

**Lab 12: Neural Networks**

**Date: 04-05-2023**

**Time: 10:00-13:00 (A) and 14:00 to 17:00 (B)**

**Instructor: Dr. Hashir Kiani**

**Lab Engineer: Ms Shakeela Bibi**

| Name : Nirmal rai  Class : BS CS 10-B  CMS : 337582 |
| --- |

# Lab 12: Neural Networks

**Introduction:**

Neural networks are computing systems vaguely inspired by the biological neural networks that constitute human brain. A neural network is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron that receives a signal then processes it and can signal neurons connected to it. The "signal" at a connection is a real number, and the output of each neuron is computed by some non-linear function of the sum of its inputs. The connections are called edges. Neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Neurons may have a threshold such that a signal is sent only if the aggregate signal crosses that threshold. Typically, neurons are aggregated into layers. Different layers may perform different transformations on their inputs. Signals travel from the first layer (the input layer) to the last layer (the output layer), with hidden layers in between the input and output layer.

**Lab Task:**

This lab comprises of two tasks.

**Task 1:** You are supposed to implement a multi-layer neural network (from scratch) to perform the logic Exclusive OR operation. You are to use the minimum possible number of neurons, at max three neurons (two hidden and one output neuron). Note that you are only allowed to use core Python functions without any additional library providing you the neural network functions.

**Task 2:** You are supposed to perform multi-class classification through neural networks. Data in excel files (both the training and test sets) are uploaded on LMS (same data you used in previous lab). In the said training and test data files, each row contains data about one instance of a plant category with four predictors/attributes. The output is one of the following classes: “Arctica”, “Harlequin” or “Caroliniana”. Please submit your results for the complete test.

Note: Feel free to use any suitable Python library for task 2.

# Task 1 :

## Code :

| import math  # Sigmoid activation function  def sigmoid(x):      return 1 / (1 + math.exp(-x))  # Derivative of the sigmoid activation function  def sigmoid\_derivative(x):      return sigmoid(x) \* (1 - sigmoid(x))  # Initialize the weights for the two layers randomly  weights1 = [[1, 1], [1, 1]]  weights2 = [1, -2]  # Define the inputs and targets  inputs = [[0, 0], [0, 1], [1, 0], [1, 1]]  targets = [0, 1, 1, 0]  # Train the network for a number of epochs  for epoch in range(5000):      # Iterate over each input/target pair      for i in range(len(inputs)):          # Forward propagation          hidden = [sigmoid(sum([inputs[i][j] \* weights1[j][k] for j in range(2)])) for k in range(2)]          output = sigmoid(sum([hidden[k] \* weights2[k] for k in range(2)]))            # Backpropagation          error = targets[i] - output          output\_delta = error \* sigmoid\_derivative(output)          hidden\_deltas = [output\_delta \* weights2[k] \* sigmoid\_derivative(hidden[k]) for k in range(2)]            # Update the weights          for j in range(2):              for k in range(2):                  weights1[j][k] += inputs[i][j] \* hidden\_deltas[k]          for k in range(2):              weights2[k] += hidden[k] \* output\_delta  # Test the network with some inputs  test\_inputs = [[0, 0], [0, 1], [1, 0], [1, 1]]  for i in range(len(test\_inputs)):      hidden = [sigmoid(sum([test\_inputs[i][j] \* weights1[j][k] for j in range(2)])) for k in range(2)]      output = sigmoid(sum([hidden[k] \* weights2[k] for k in range(2)]))      print(test\_inputs[i], output) |
| --- |

## Screenshot :

Text

Description automatically generated

# Task 2

## Code :

| import pandas as pd  import numpy as np  import tensorflow as tf  from sklearn.model\_selection import train\_test\_split  from sklearn.preprocessing import LabelEncoder  train\_data = pd.read\_excel('/content/TrainingSet.xlsx')  print(train\_data)  X = train\_data.iloc[:, :-1].values  y = train\_data.iloc[:, -1].values  # Convert labels to numerical values  le = LabelEncoder()  y = le.fit\_transform(y)  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  model = tf.keras.models.Sequential([  tf.keras.layers.Dense(32, activation='relu', input\_shape=(4,)),  tf.keras.layers.Dense(16, activation='relu'),  tf.keras.layers.Dense(3, activation='softmax')  ])  model.compile(optimizer='adam',  loss='sparse\_categorical\_crossentropy',  metrics=['accuracy'])  history = model.fit(X\_train, y\_train, epochs=100, validation\_data=(X\_test, y\_test))  test\_loss, test\_acc = model.evaluate(X\_test, y\_test)  print('Test loss:', test\_loss)  print('Test accuracy:', test\_acc) |
| --- |

## Screenshot :



