EE30241 Coursework – Activity 3b
Multilayer network and digit recognition

Uriel Martinez-Hernandez

November 2021

Introduction

You will implement a 2-layer Multilayer Neural Network or Multiplayer Perceptron (MLP) for the recognition input bits from a 2-bit XOR gate. This MLP will be composed of an input, a hidden and an output layer. You will implement this network in Python without any machine learning library. You will also implement an MLP for recognition of digits from 0 to 9 and communication via ROS. To get started, download the zip file *'Files for CW3b-multilayer_network_and_digit_recognition'* from Moodle, where you will find the following:

1. **MLP_with_backpropagation.py:** Example of Python program with the random initialisation of weights and bias for a 2-layer MLP. You will need to implement the complete algorithm based on the lecture notes.

2. **MLP_mnist_digit_recognition.py:** An example of MLP network for the recognition of digits from the MNIST dataset. This Python program is implemented using TensorFlow and Keras libraries.

3. **MLP_Bath_digit_recognition.py:** Example of Python program with functions to load digits from the Bath dataset and prepare the training and testing data. You will need to complete this program for the recognition of digits from 0 to 9 following the example code (MLP_mnist_digit_recognition.py) from previous point 2. The link to download the Bath dataset is available on Moodle.

Activity 1 – XOR input pattern recognition with 2-layer MLP and backpropagation

Implement a 2-layer MLP network using backpropagation to recognise the input bits from an XOR gate. Follow the algorithm from the lecture notes and implement it in the file *MLP_with_backpropagation.py* that has been provided for this activity. Figure 1 shows the description of the MLP network and input data and labels that you will use in activity 1. Note that the input layer only receives the input bits from X1 and X2 but does not perform any processing step. The hidden and output layer process the input signals together with the weights and bias to generate the output value Y.

Table 1 – Input data and labels

X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	0

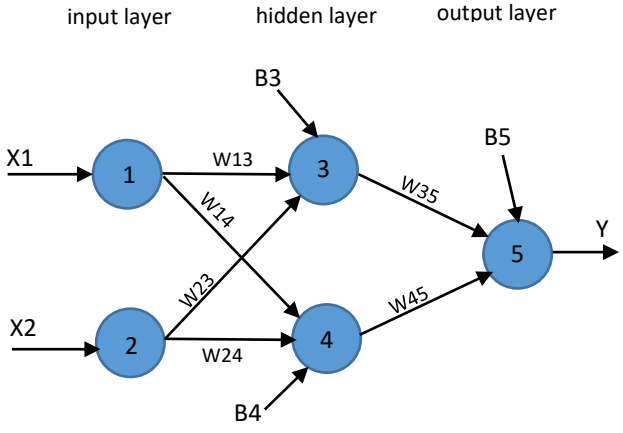

Figure 1 – 2-layer MLP

Activity 2 – Open, run and explore the MLP MNIST program

Open and run the program *MLP_mnist_digit_recognition.py*, which trains a 3-layer MLP for the recognition of digits from the MNIST dataset. It is recommended to open this program using Spyder in a conda environment. Run this program and see the output accuracy for recognition of the training and testing datasets.

Have a look at the functions to create an MLP network using TensorFlow and Keras libraries. The input data (images from digits) are flattened to be able to connect them to the neurons of the MLP network.

The input and hidden layers have 50 neurons each and sigmoid activation function. The output layer has 10 neurons which correspond to the 10 digits in the dataset, and it uses the softmax function to get the probability of the prediction. The parameters of the network are defined as follows: learning rate=0.001, decay=1e-7 and momentum=0.9.

The training process of the MPL is performed using 50 epoch and an input image batch of 64. Have a look at the syntax of the functions to fit the model and make predictions once the model has been trained.

Activity 3 – Digit recognition using TensorFlow, Keras and the Bath dataset

In this activity, you will recognise digits from 0 to 9 using the Bath dataset. You will develop a Python program to implement an MLP network using TensorFlow and Keras. The template program *MLP_Bath_digit_recognition.py* with the functions to load the training and testing datatset has been provided to you. You need to complete this program to recognise the digits following the logic used in the program *MLP_mnist_digit_recognition.py* from Activity 2. If your program does not achieve accurate recognition in both the training and testing datasets, then you need to adjust the parameters (e.g., learning rate) and hyper-parameters (e.g., adding more neurons and layers to the network).

Activity 4 – Communication of digit recognition with calculator via ROS

Once you completed Activity 3, create a copy of your file *MLP_Bath_digit_recognition.py* and save it with the name *MLP_Bath_digit_recognition_ROS.py*. In this new file, you will need to include the require ROS code to publish the predicted digit from your system in order to be received by the calculator (developed with ROS) that you developed in the Coursework Activity 2 (Part 1 of EE30241 unit).

Let's suppose that you want to add two numbers A + B, then your calculator should work as follows:

- The first number A will be provided by the user
- The operator + will be provided by the user
- The second number B will be recognised and published by your MLP and received by your calculator via ROS.

This means that you will need to create a publisher in your Python program with your MPL network and you will need to have a subscriber in your calculator. Figure 2 illustrates an example of this process using ROS nodes, you can decide the name of the nodes, publishers and subscribers.

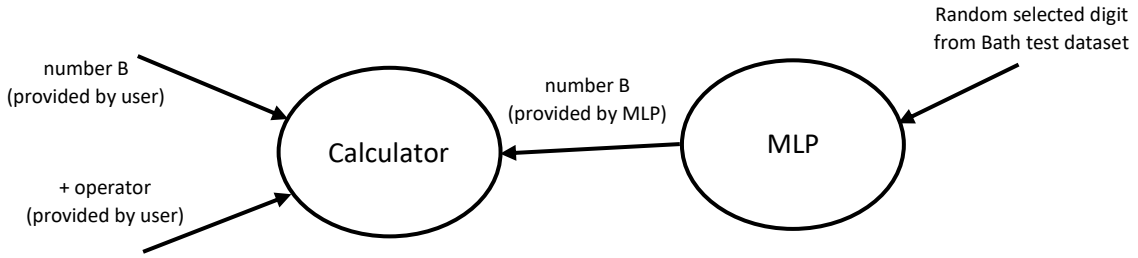

Figure 2 – Illustration of the communication between the Calculator and MLP network

Activity 5 – Record a video of the calculator and MLP network working together

Run your calculator and your MLP network. Open a random digit from the testing dataset of the Bath data, show in a figure (e.g., imshow) the digit randomly selected and display on the terminals the number recognised by your MLP network, the number received by your calculator and the result of the operation performed by your calculator. You can select what operation to perform (e.g., +, -, /) and repeat the complete process for different numbers from 0 to 9 randomly selected. Record a short video (~2 minutes) of this process and name the file as *'Calculator_digit_recogniser.mp4'*.

Files that you will need to submit from this activity:

- MLP_with_backpropagation.py
- MLP_Bath_digit_recognition.py
- MLP_Bath_digit_recognition_ROS.py
- Calculator_digit_recogniser.mp4

IMPORTANT

Add clear comments to your files and remember that it is important to indent your code to make it easier to read by others. All these aspects will be considered for your mark.