PMDS603P Deep Learning Lab Experiment 1

July 2025

1 Work to do today

Note: Make a single PDF file of the work you are doing in Jupyter notebook. Upload with the proper format. Please mention your name and roll no properly with the Experiment number on the first page of your submission.

Q1. Using McCulloch Pitts model discussed in the class, write a Python code to implement the **OR** and **AND** Boolean functions. Also, plot the boundary input points and the linear classifier that we get in that case.

Q2. Write a Python code to implement the Perceptron Learning Algorithm to implement OR, AND, NAND, NOR logic Gates and report the weights and bias. Also, print the inputs and outputs after training the weights properly. Further plot the linear classifier that you have obtained as well. Note that here the input vectors x should be extended with a x_0 term, which is taken as 1. Assume we have two inputs x_1 and x_2 so the $\mathbf{x}$ vector would be $[1, x_1, x_2]$

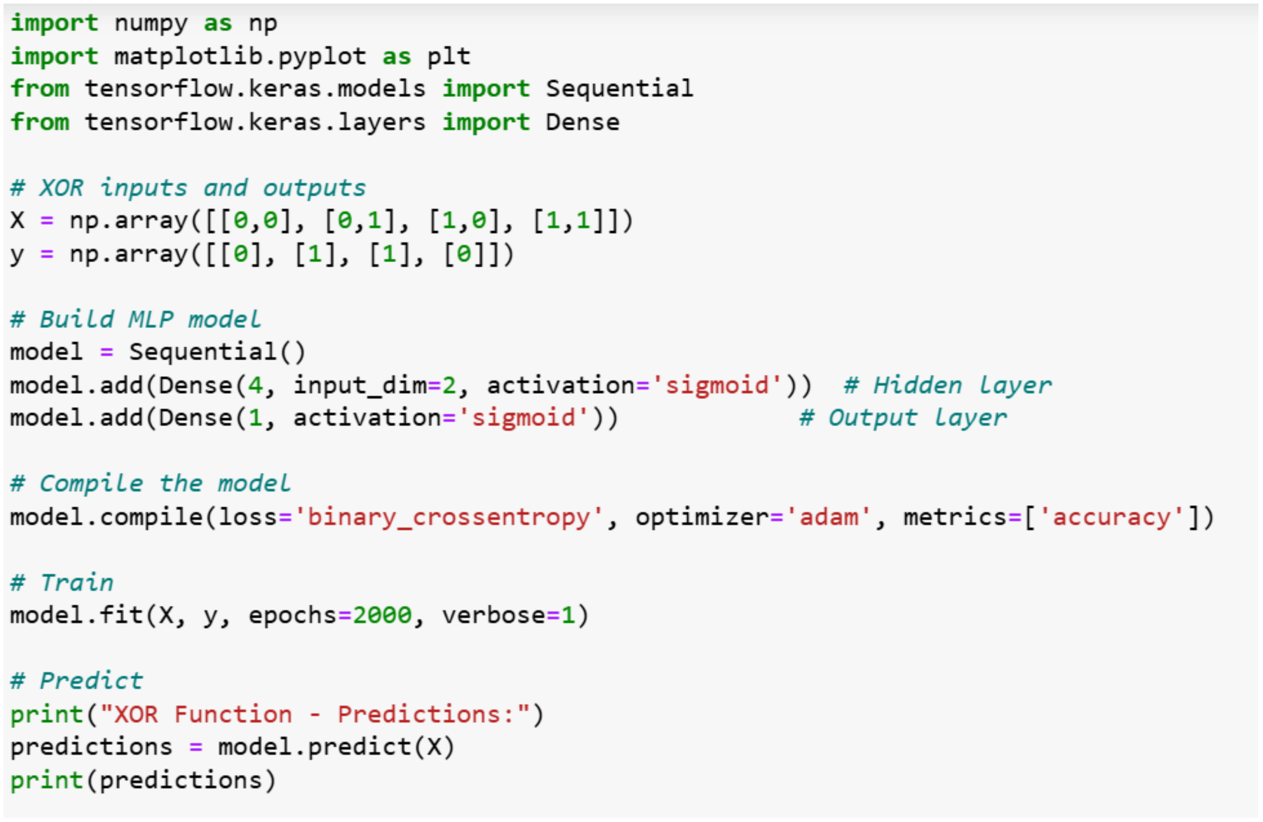
Perceptron training algorithm

Algorithm 1 Perceptron Learning Algorithm

```
$P \leftarrow$ inputs with label 1
$N \leftarrow$ inputs with label 0
Initialize $\mathbf{w} = [w_0, w_1, \ldots, w_n]$ randomly
while !convergence do
  Pick random $\mathbf{x} \in P \cup N$
  if $\mathbf{x} \in P$ and $\sum_{i=0}^{n} w_i \cdot x_i < 0$ then
    $\mathbf{w} = \mathbf{w} + \mathbf{x}$
  end if
  if $\mathbf{x} \in N$ and $\sum_{i=0}^{n} w_i \cdot x_i \geq 0$ then
    $\mathbf{w} = \mathbf{w} - \mathbf{x}$
  end if
end while
```

The algorithm converges when all the inputs are correctly classified.

Q3. Next we will see how we can implement an XOR function using inbuild tools. Here



```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# XOR inputs and outputs
X = np.array([[0,0], [0,1], [1,0], [1,1]])
y = np.array([[0], [1], [1], [0]])

# Build MLP model
model = Sequential()
model.add(Dense(4, input_dim=2, activation='sigmoid'))  # Hidden layer
model.add(Dense(1, activation='sigmoid'))             # Output layer

# Compile the model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train
model.fit(X, y, epochs=2000, verbose=1)

# Predict
print("XOR Function - Predictions:")
predictions = model.predict(X)
print(predictions)
```

Figure 1: Enter Caption

you can see that we use these tools Keras and tensorflow, developed by Google. Sequential is a class that can be used to build deep neural networks. Here in the code the model is an object of the class sequential. We can add new layers in the neural network. the first hidden layer you see contains 4 neurons and the input dimension is also specified. In our case we have a 2-dimensional vector as input. We do not represent a separate input layer here. We are having an output layer with one neuron here. This outputs the probability that the input vector belongs to a particular class. So after compiling the model and fitting the model when we go for the predictions here we are getting nothing but the probability that the input belongs to class 1. If we get a value less than 0.5 we will have a prediction that the input belongs to class 0 and otherwise to class 1.