hw_Feed-forward Neural Networks

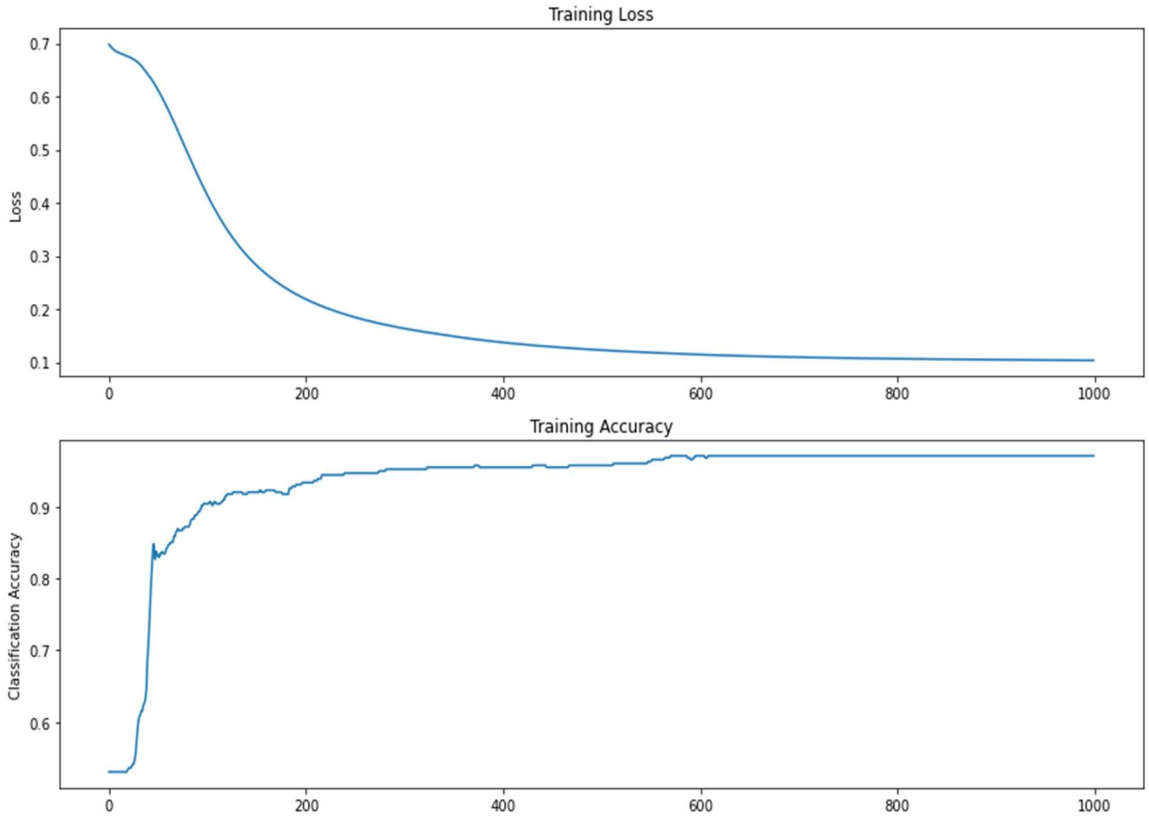
Submitted by: Satvik Kishore, Deekshita Saikia

Q1 Build a multi-layer neural network to solve the XOR classification problem. Use the provided FFNN as a starting point. Test against the data generated by gen_xor() from gen_data.py. Show:

a) the testing percent correct;
b) the training progress curves; and
c) the decision surface with overload training data.

Solution: We fit a neural network with 1 hidden layers and 5 nodes. The hidden layer uses a ReLu activation function, whereas the output layer uses a Sigmoid as the activation function. We observe the following:

a) The testing accuracy is **93.60%**.
b) The training accuracy and loss curves through epochs look as shown below:



c) The decision surface looks as shown below:

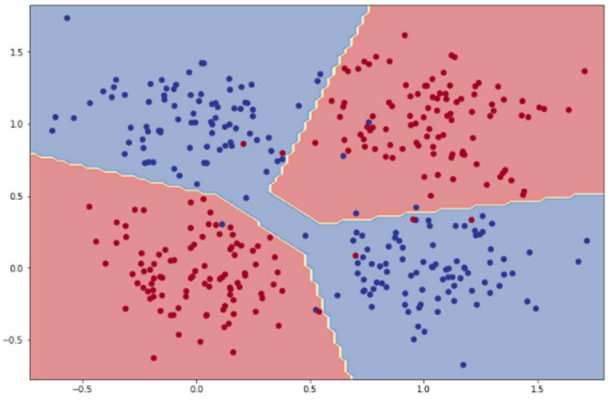


Figure 1: Decision surface with training data

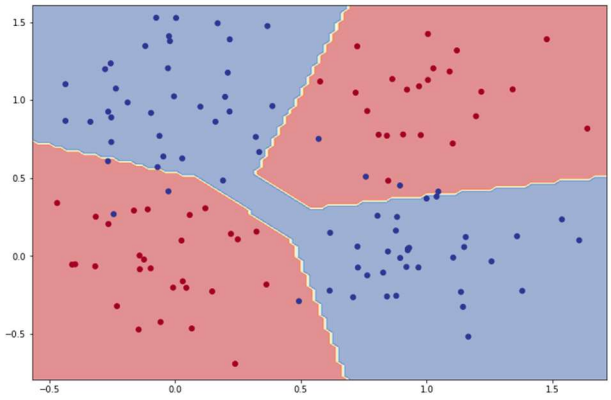


Figure 2: Decision surface with testing data

Q2 Implement a feed-forward neural network function from scratch, using only built-in Python modules and numpy. Extract the learned weights from Q1 and run the model through your custom implementation. Demonstrate that you get the same results.

Do not train the model yourself. Do not implement backpropagation. Just run it forward using the PyTorch-trained weights.

Solution: We extract the weights and biases for both layers from Q1 and save the predictions from the PyTorch model. We replicate the neural net from Q1 with numpy and compare the predictions from our custom implementation against the outputs from Q1, using **cosine similarity** as our metric. We obtain a cosine similarity value of **0.995** , indicating the angle between the two vectors is close to zero, and are hence, the same vector.