Assignment 2: Neural Networks
MA-INF 2313: Deep Learning for Visual Recognition

Theoretical Task Due Date:	07.11.2019
Programming Task Due Date:	14.11.2019
Assistant:	Soumajit Majumder

1 Theoretical Exercises (15 pts)

1. **(6 pts)** Neural networks can be interpreted geometrically as a partitioning of the input feature space $\boldsymbol{x}$. Now consider the trapezoid in Figure 1, which represents a decision boundary, where the region inside belongs to class A and the region outside belongs to class B. How should one design a two-layered network[1] to represent this trapezoid? Fully specify your network, including the form of input, hidden and output activation functions as well as associated parameters.

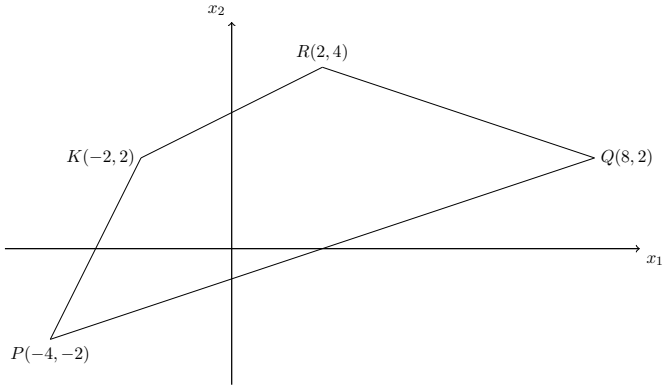

Figure 1: The region in the trapezoid belongs to class A and the other region to class B.

2. **(9 pts)** Consider the Multi-Layer Perceptron with a single hidden-layer given in Figure 2 with two input units x_1, x_2, two hidden units h_1, h_2, two output units o_1, o_2 and additionally, the hidden and output bias units b_1, b_2.

   In this task you are going to work with a single training set: given inputs 0.1 and 0.4, we want the neural network to output 0.1 and 0.9. You will use the Sigmoid function as your activation function for each hidden layer units h_1, h_2 and output units o_1, o_2.

   Run a forward pass to see if the neural network with the current weights predicts the outputs correctly. Show your work step by step for the forward pass. Calculate the total error in the network using mean squared error.

[1] as per Bishop's notation, which results in one input layer, one hidden layer and one output layer.

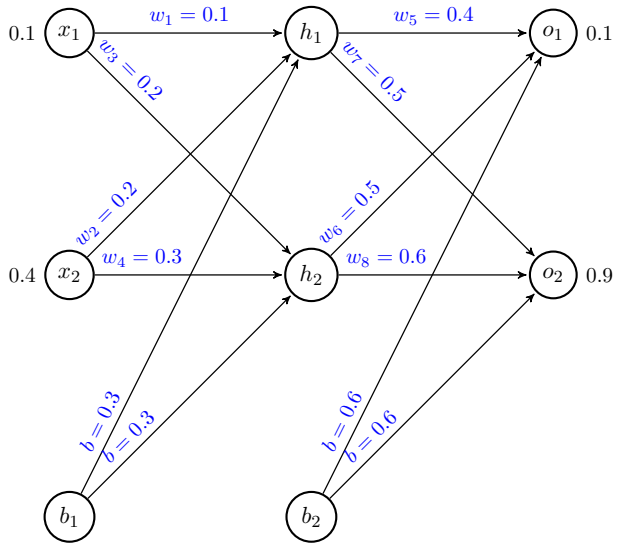

Figure 2: Multi-Layer Perceptron with a single hidden-layer

2 Programming Exercises (15 pts)

In the programming part, you will set up, train and test a Multi-Layer Perceptron for classification **using any library of your choice**. Write the code for a Multi-Layer Perceptron (MLP) with two hidden layers (feel free to pick the number of hidden units per layer) and test your classifiers on the following two datasets. For the loss function, you can use the cross-entropy loss. Feel free to experiment with other forms of classification losses as well.

1. **(6 pts)** First, classify hand-written digits on the MNIST dataset, according to the given training and testing splits on the website. Report the final classification accuracy on the test set. Additionally, plot the training loss over the iterations.

2. **(9 pts)** For the second part, train a classifier to identify and distinguish between 20 different people. The dataset is a subset of The ORL Database of Faces and is provided in the file `ORL_faces.npz.zip`. The size of each image is 92×112 pixels, with 256 grey levels per pixel. It is recommended to visualize the dataset as a sanity check. You can use the provided `load_ORL_faces.py` to load the data. Train your network with the provided training (trainX, trainY) split and report the classification accuracy for the test (testX, testY) split. Report the final test accuracy and plot the training loss over the iterations.