EEE 443/543: Neural Networks

Assignment 1

(Due 05 November 2021, 17:00PM)

Instructions:

1. Prepare a report (including your answers/plots) to be uploaded on Moodle.
2. The report should be typeset (no handwriting allowed except for lengthy derivations, which may be scanned and embedded into the report).
3. Show all steps of your work clearly.
4. Unclear presentation of results will be penalized heavily.
5. No partial credits for unjustified answers.
6. **Use of any toolbox or library for neural networks is prohibited**.
7. Return all Matlab/Python code that you wrote in a single `.m/.py` file.
8. Code should be commented, code for different HW questions should be clearly separated.
9. The code file should NOT return an error during runtime.
10. If the code returns an error at any point, the remaining part of your code will not be evaluated (i.e., 0 points).

Question	Points	Your Score
Q1	20	
Q2	25	
Q3	35	
Q4	20	
TOTAL	100	

Question 1. [20 points]

A single neuron receives input from m input neurons with weights w_i, where $i \in [1\ m]$. The neuron is expected to predict the probability that the output t belongs to *Class A* ($t = 1$) versus *Class B* ($t = -1$). A datasets of training samples are available with inputs x^n and outputs y^n ($n \in [1\ N]$). You are told that the maximum a posteriori estimate for the network weights are obtained by solving the following optimization problem:

$$\arg\min_{W} \sum_{n} \left(y^n - h(x^n, W)\right)^2 + \beta \sum_{i} w_i^2 \tag{1}$$

where W is the vector of weights w_i, β is a scalar constant, and $h(.)$ is the output of the neuron. According to this estimate, derive the prior probability distribution of the network weights analytically.

Question 2. [25 points]

An engineer would like to design a neural network with a single hidden layer with four input neurons (with binary inputs) and a single output neuron to implement:
(X_1 OR NOT X_2) XOR (NOT X_3 OR NOT X_4)
Assume a hidden layer with four hidden units, and a unipolar activation function (i.e., the step function). Answer the questions below.

a) For each hidden unit, analyically derive the set of inequalities based on which a set of weights and an activation threshold can be selected.

b) Choose a particular weight vector (including the bias term), and show that the designed network achieves 100% performance in implementing the desired logic.

c) Now assume that the input data samples are subject to small random fluctuations due to noise. Will the network you designed in **part a** function robustly under noisy conditions? Find the set of weights and the activation threshold for the most robust decision boundary.

d) Generate 100 input samples by first concatenating 25 samples from each input vector. Generate a random noise vector of length 2 for each training sample, assuming a zero-mean Gaussian distribution with an std of 0.2. Form validation samples for testing the NNs by linearly superposing the input samples and the random noise samples. Evaluate the classification performance (i.e., percentage correct) of the networks designed in **parts a and c** on the validation samples. Interpret your results.

Question 3. [35 points]

A researcher would like to process images of alphabet letters with a perceptron. A collection of images were compiled for training and testing the perceptron. The file `assign1_data1.h5` contains variables `trainims` (training images) and `testims` (testing images) along with the ground truth labels in `trainlbls` and `testlbls`. Answer the questions below.

a) Visualize a sample image for each class. Find correlation coefficients between pairs of sample images that you have selected. Display the correlations in matrix format. Discuss the degree of within-class versus across-class variability.

b) Design a single-layer perceptron with an output neuron for each digit, using the training data. Set the initial network weights w and bias term b as random numbers drawn from a Gaussian distribution $\mathcal{N}(0, 0.01)$, assume a sigmoid activation function. Your implementation should not train each output neuron separately, but a compound matrix W and a compound vecor b should be defined and used to simultaneously update all connections. The online training algorithm should perform 10000 iterations. At each iteration, a sample image should be randomly selected from the training data, the network should be updated according to the gradient-descent learning rule, and W, b, and the mean-squared error (MSE) should be recorded. Tune the learning rate η^* in order to minimize the final value of the MSE. Display the final network weights for each digit as a separate image, and describe the visual characteristics.

c) Now separately repeat the training process using a substantially higher and a subtantially lower value thant η^*. On a single figure, plot the MSE curves (across all 10000 iterations) for η_{high}, η_{low} and η_*. Discuss your results.

d) Validate the performance of the trained networks using all samples in the test data. Report the performance values for the three networks with η_{high}, η_{low} and η_*.

Question 4. [20 points]

The goal of this question is to introduce you simple two-layer neural networks, and to let you examine the effects of various hyperparameter selections on these classical model. You will be experimenting with a Python demo on a network model. Download `demo_tln.zip` from Moodle and unzip it. The demo is given asa Jupyter Notebook along with relevant code and data. The easiest way to install Jupyter with all Python and related dependencies is to install Anaconda. After that you should be able to run through the demo in your browser easily. The point of this demo is that it takes you through the training algorithms step by step, and you need to inspect the relevant snippets of code for each step to learn about implementation details.

The notebook `two_layer_net.ipynb` contains demonstrations on a simple two-layer network model. You need to run the demo till the end without any errors. You may have to debug the code in case of any fatal errors. You are supposed to convert the outputs of the completed demo to a PDF file, and attach it to the project report. You should also comment on your results and answer any inline questions that are provided in the notebook.