Programming Assignment 2: Recognition of Handwritten Digits by Multilayer Perceptrons

You are given a zipped data set (Data.rar) containing four files:

1. in_train.txt: the input features of training data (7494 handwritten digits, with each having 16 features which are the (x,y) coordinates of 8 points sampled sequentially from the writing trajectory of the digit);
2. out_train.txt: the desired class label, i.e. 0~9, corresponding to each sample in in_train.txt;
3. in_test.txt: the input features of testing data (3498 handwritten digits, with each having 16 features which are the (x,y) coordinates of 8 points sampled in sequence during writing);
4. out_test.txt: the desired class label, i.e. 0~9, corresponding to each sample in in_test.txt.

The data set is from http://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits. You have to train your MLP with the in_train.txt and out_train.txt. Note that the data may need to be preprocessed as follows:

1. Since a sigmoid activation function has the derivatives almost equal to 0 at both extreme ends ($x \to \infty$ and $x \to -\infty$), a neuron employing the sigmoid activation function would become "dull" or even "dead" (no learning capability) for both very large values and very small values of x. In view of this, you have to normalize the x's and y's of the coordinates to be within the range [0,1] to prevent the neurons from dullness. Besides, this normalization also remove the size variations of the written digits.
2. Remember the one-output-one-class architecture of the MLP I taught you in our class. Therefore, you have to prepare the appropriate form for the target data in the out_train.txt and out_test.txt.

For your convenience, I list the backpropagation learning algorithm of the MLP in the following.

Backpropagation Algorithm:

Input: L*:number of layers*, $\{S^{(l)}\}_{l=1}^{L}$*:number of neurons in layer* l, $f^{(l)}$*: the activation function of each neuron in layer l for l=1,…,L,, and*
α: *learning rate (a small constant)*

Output: $$\mathrm{W}^{(l)} = \begin{bmatrix} \mathbf{w}_1^{(l)} \\ \vdots \\ \mathbf{w}_{S^{(l)}}^{(l)} \end{bmatrix} = \begin{bmatrix} w_{1,1}^{(l)} & \cdots & w_{1,S^{(l-1)}}^{(l)} \\ \vdots & \ddots & \vdots \\ w_{S^{(l)},1}^{(l)} & \cdots & w_{S^{(l)},S^{(l-1)}}^{(l)} \end{bmatrix},\ \mathbf{b}^{(l)} = \begin{bmatrix} b_1^{(l)} \\ \vdots \\ b_{S^{(l)}}^{(l)} \end{bmatrix}, \text{ for } l = 1, \dots, L$$

Steps:

1. **Input a set of training examples**
2. **For each training example** $\{\mathbf{t}, \mathbf{p}\}$ **:** Set the corresponding input activation $\mathbf{a}^{(0)} = \mathbf{p}$, and perform the following steps:
   - **Feedforward:** For each $l = 1, 2, \dots, L$ compute $\mathbf{n}^{(l)} = \mathrm{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$ and $\mathbf{a}^{(l)} = f(\mathbf{n}^{(l)})$, where $f(\cdot)$ denotes the activation function.
   - **Output layer error** $\boldsymbol{\delta}^{(L)} = -2F'(\mathbf{n}^{(L)})(\mathbf{t} - \mathbf{a}^{(L)})$, where $F'(\mathbf{n}^{(L)}) = \begin{bmatrix} f'(n_1^{(L)}) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & f'(n_{S^L}^{(L)}) \end{bmatrix}$ where $f'()$ denotes first derivative of the activation function. For example, the first derivative of a sigmoid function can be computed by $f' = f(1 - f)$.
   - **Backpropagate the error:** For each $l = L-1, L-2, \dots, 1$ compute $\boldsymbol{\delta}^{(l)} = F'(\mathbf{n}^{(L)}) \cdot ((\mathrm{W}^{(l+1)})^T \boldsymbol{\delta}^{(l+1)})$.
   - **Calculate weight and bias updates**: For each $l = L-1, L-2, \dots, 1$ compute $\Delta\mathbf{w}^{(l)} = \boldsymbol{\delta}^{(l)}(\mathbf{a}^{(l-1)})^T$ and $\Delta\mathbf{b}^{(l)} = \boldsymbol{\delta}^{(l)}$.
   - **Update weights:** For each $l = L-1, L-2, \dots, 1$ update the weights and biases according to the rule $\mathbf{w}^{(l)} \to \mathbf{w}^{(l)} - \alpha\Delta\mathbf{w}^{(l)}$ and $\mathbf{b}^{(l)} \to \mathbf{b}^{(l)} - \alpha\Delta\mathbf{b}^{(l)}$.

Please do not use also the In_test.txt and Out_test.txt as your training data because these two files should be used only for evaluating the performance of your MLP. You are required to try at least 5 MLPs with different network architectures (i.e., number of hidden layers and number of hidden neurons) to examine their impacts on the performance.

Files you should submitted to the course website (www.elearn.ndhu.edu.tw):

1. Source codes with concise comments.

2. A Word (or PDF) report which presents your results as clearly as possible.
   Some required contents in your report include
   - the extra efforts (such as extracting other effective features, designing friendly GUI for demo, ...), if any, you have made in this assignment,
   - the tables that lists the recognition rates of different network architectures,
   - the confusion matrix (please search on google for the definition of a confusion matrix),
   - the detailed discussions on your results, and
   - the conclusive descriptions about what you have learned in this assignment (as many as possible) .

If you get any questions when doing this assignment, feel free to post your questions to our class forum on course website.