EE5904R/ME5404 Neural Networks: Homework #2

Important note: the due date is 05/03/2015. You may choose to hand in your answer sheets during the break of the lecture, or directly to the teaching assistant in the lab. Late submission is not allowed unless it is well justified. Also only hardcopy will be accepted. Softcopy by email will not be allowed unless with valid reasons. Please supply the MATLAB code in your answer if computer experiment is involved.

Q1. **Rosenbrock's Valley Problem (10 Marks)**

Consider the Rosenbrock's Valley function:

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2$$

which has a global minimum at (x,y) = (1,1) where f(x,y) = 0. Now suppose the starting point is randomly initialized in the open interval (0, 1) for x & y, find the global minimum using:

a). Steepest (Gradient) descent method

$$w(k+1) = w(k) - \eta g(k)$$

with learning rate η = 0.001. Record the number of iterations when f(x,y) converges to (or very close to) 0 and plot out the trajectory of (x,y) in the 2-dimensional space. Also plot out the function value as it approaches the global minimum. What would happen if a larger learning rate, say η = 0.1, is used?

(5 Marks)

b). Newton's method (as discussed on page 13 in the slides of lecture Four)

$$\Delta w(n) = -H^{-1}(n) g(n)$$

Record the number of iterations when f(x,y) converges to (or very close to) 0 and plot out the trajectory of (x,y) in the 2-dimensional space. Also plot out the function value as it approaches the global minimum.

(5 Marks)

Q2. Function Approximation (20 Marks)

Consider using MLP to approximate the following function:

$$y = 1.2\sin(\pi x) - \cos(2.4\pi x) \quad \text{, for} \quad x \in [-1,1].$$

The training set is constructed by dividing the range [-1, 1] using a uniform step length 0.05, while the test set is constructed by dividing the range [-1, 1] using a uniform step length 0.01. You may use the MATLAB neural network toolbox to implement a MLP (see the Appendix for guidance) and do the following experiments:

a). Use the sequential mode with BP algorithm and experiment with the following different structures of the MLP: 1-n-1 (where n = 1,2,...,10,50). For each architecture plot out the outputs of the MLP for the test samples after training and compare them to the desired outputs. Try to determine whether it is under-fitting, proper fitting or over-fitting. Identify the minimal number of hidden neurons from the 11 experiments, and check if the result is consistent with the guideline given in the lecture slides. Compute the outputs of the MLP when x=-1.5 and 1.5, and see if the MLP can make reasonable predictions outside of the range of the input limited by the training set.

(7 Marks)

b). Use the batch mode with trainlm algorithm to repeat the above procedure.

(7 Marks)

c). Use the batch mode with trainbr algorithm to repeat the above procedure.

(6 Marks)

Q3. Facial Image Based Glasses Wearing Recognition (30 Marks)

Multi-layer perceptron (MLP) can be used to solve real-world pattern recognition problems. In this assignment, the following tasks are involved: applying the MLP to a two-class problem: glass wearing recognition based on face images. You may download the zipped file containing the database from IVLE. And after unzipping, you may find two folders: Train and Test. The training set consists of 100 face images from 50 with-glasses persons and 50 without-glasses persons while the test set consists of 30 face images from 15 with-glasses persons and 15 without-glasses persons.

The following function in MATLAB can help you to read images from graphics file:

A = imread(<u>filename</u>);

where the string <u>filename</u> specifies a grayscale or color image. The return value A is an array containing the image data. If the file contains a grayscale image, A is an M-by-N array. If the file contains a true color image, A is an M-by-N-by-3 array. For simplicity, grayscale images are used in this assignment. Please use the function *dir()* to generate the filenames inside a folder for code efficiency.

For example,

A = imread('Test/WithGlasses/072.jpg');

this code will read the image 072.jpg in test set into MATLAB. After reading a image into an array, you could display the grayscale image A using the following function in MATLAB:

imshow(A,[]);

In order to efficiently process all the image data, you may need to convert the matrix form data A to a vector form using the following code:

B = A(:);

and the resulting B is a column vector whose elements are taken column-wise from A. You could group all the training images together using Train_C = [B1, B2, ..., B100] and all the test images together using Test_C = [Bt1, Bt2,...,Bt30]. In the next, these vectors are used as the input data to the neural networks.

a). Apply the Rosenblatt’s perceptron (single layer perceptron) to the glass-wearing recognition problem. After the training procedure, calculate the recognition errors for both the training set and the test set, and evaluate the performance of the network.

(10 Marks)

b). Apply the Multi-layer perceptron to the glass-wearing recognition problem using batch mode of learning. After the training procedure, calculate the recognition error for both training set and test set, and evaluate the performance of the network.

(10 Marks)

c). Apply the Multi-layer perceptron to the glass-wearing recognition problem using sequential mode of learning. After the training procedure, calculate the recognition error for both training set and test set, and evaluate the performance of the network. Which one is the best in terms of test error?

(10 Marks)

Important note: There are many design and training issues to consider when you apply neural networks to solve real world problems. We have discussed most of them in the lecture four. Some of them have clear answers, some of them may rely on empirical rules, and some of them have to be determined by trial and error. I believe that you will have more fun playing with these design parameters and making your own judgment rather than solving the problem with a prescribed set of parameters. Hence, there is no standard answer to this problem, and the marking will be based upon the whole procedure rather than the final recognition accuracy.

Appendix

1. Create a feed-forward back propagation network using MATLAB toolbox:

```
net = newff(minmax(X),[S1 S2...SNl],{TF1 TF2...TFNl}, BTF, PF)
```

where the arguments are specified as follows:

X – The matrix which stores all the input samples with the number of columns being the number of samples, and the number of rows being the dimension of the input vector.

Si -- Size of ith layer, for Nl layers

TFi -- Transfer function of ith layer

BTF -- Backpropagation network training function

PF -- Performance function, default = 'mse' and returns an N layer feed-forward backprop network.

The transfer functions TFi can be any differentiable transfer function such as tansig, logsig, or purelin. The training function BTF can be any of the backprop training functions such as trainlm, traincgf, traingd, traingdm etc.

For example, below is a problem consisting of inputs P and targets T to be solved with a network.

```
P = [0 1 2 3 4 5 6 7 8 9 10];
T = [0 1 2 3 4 3 2 1 2 3 4];
```

In the following a two-layer feed-forward network is created. The first layer has five tansig neurons, the second layer has one purelin neuron. The trainlm network training function is to be used.

```
net = newff(minmax(P),[5 1],{'tansig' 'purelin'},'trainlm');
```

**Note:*
1). Please use "traincgf" instead if you encounter "out of memory" problem when using "trainlm".
2). In the MATLAB 2008 and higher version, the function newff() can be used in the following way:

```
net = newff(P,T,5,{tansig' 'purelin'},'trainlm');
```

You may choose either of them to conduct the experiment.
Then the network is simulated and its output plotted against the targets as follows.

```
Y = sim(net,P);
plot(P,T,P,Y,'o')
```

Finally the network is trained for 50 epochs, and then the network's output is plotted.

```
net.trainParam.epochs = 50;
net = train(net,P,T);
Y = sim(net,P);
plot(P,T,P,Y,'o')
```

You are recommended to play with the MATLAB code on page 33 in the lecture notes of lecture Four to get familiar with the neural network toolbox.

2. The train function used above provides batch learning mode only. In order to enable the sequential learning mode, the following strategy is needed:
Set the number of epoch of batch training to be 1, and perform the sequential leaning step by step.
For example, for the same problem mentioned above, the following code will enable the sequential learning:

```
epoch_s    = 50;       % Specify the number of epoch for sequential training
net.trainParam.epochs    = 1;                  % Set epoch of batch training to be 1
for i = 1 : epoch_s
       index = randperm(length(P))             % Shuffle the input data every epoch
       net = adapt(net,P(index),T(index));    % Perform sequential learning
end
```

**Note:*
1). randperm() and length() are built-in functions in MATLAB. Please type "doc randperm" and "doc length" to find detailed information of these two functions.
2). For sequential learning, first order training method such as "traingd" and "traingdm" is sufficient, so there is no need to use second order training method such as "trainlm" and "traincgf".

3. After training, the weights in the hidden neurons are stored in the structure "net". That is, net.layerWeights. This property holds structures of properties for each of the network's layer weights. It is always an N1×N1 cell array, where N1 is the number of network layers (net.numLayers). The structure defining the properties of the weight going to the ith layer from the jth layer (or a null matrix []) is located at

net.layerWeights{i,j} if net.layerConnect(i,j) is 1 (or 0). For example, for the same problem mentioned above, after training, type "net" in the command line of MATLAB, you may obtain the following message:

```
net =

    Neural Network object:

    architecture:
             numInputs: 1
             numLayers: 2
           biasConnect: [1; 1]
          inputConnect: [1; 0]
          layerConnect: [0 0; 1 0]
         outputConnect: [0 1]
            numOutputs: 1    (read-only)
        numInputDelays: 0    (read-only)
        numLayerDelays: 0    (read-only)

    subobject structures:
                inputs: {1x1 cell} of inputs
                layers: {2x1 cell} of layers
               outputs: {1x2 cell} containing 1 output
                biases: {2x1 cell} containing 2 biases
          inputWeights: {2x1 cell} containing 1 input weight
          layerWeights: {2x2 cell} containing 1 layer weight

    functions:
              adaptFcn: 'trains'
             divideFcn: (none)
           gradientFcn: 'gdefaults'
               initFcn: 'initlay'
            performFcn: 'mse'
              plotFcns: {'plotperform','plottrainstate','plotregression'}
              trainFcn: 'trainlm'

    parameters:
            adaptParam: .passes
           divideParam: (none)
         gradientParam: (none)
             initParam: (none)
          performParam: (none)
            trainParam: .show, .showWindow, .showCommandLine, .epochs,
                        .time, .goal, .max_fail, .mem_reduc,
                        .min_grad, .mu, .mu_dec, .mu_inc,
                        .mu_max

    weight and bias values:
                     IW: {2x1 cell} containing 1 input weight matrix
                     LW: {2x2 cell} containing 1 layer weight matrix
                      b: {2x1 cell} containing 2 bias vectors

    other:
                  name: ''
              userdata: (user information)
```

You may use net. layerWeights or net.LW to check the detailed values of weights. Similarly, you may use net.biases and net.b to check the detailed values of biases.

Besides, all the information of the trained network is stored in the structure "net". You may type "doc" command to open the help manual and search for "net" (network properties) to find more details.

4. Different training algorithms have different parameters. Usually, the default values of these parameters are suitable for the experiments except the number of epochs (net.trainParam.epochs). However, you may type "doc TRAINING_ALGORITHM_NAME" (i.e., "doc trainlm") to check the parameter list of certain training algorithm.

5. The following code can be used to measure the computing time of your program :

```
tic
    Your own code here
time_t = toc;
```

6. Note that since glasses-wearing recognition is a two-class problem, proper threshold is needed after training and test. For example, if logsig is used in the output layer, those output values which are greater than 0.5 should be set to be 1 while others should be set to be 0. To this end, the classified target values are obtained for both training and test set. In the next, the final recognition accuracy could be obtained by comparing the classified target and the true target. For glasses-wearing recognition, you may use "traincgf" algorithm as it requires less memory.

7. If you have any doubt regarding any command, please type

help command_name

or

doc command_name

to get more information.