Biological Neural Computation

Homework problem set 2

Spring 2019

Data Assigned: **2/13/2019**
Data Due: **3/06/2019**

General Guidelines: The homework solutions should include figures that clearly capture the result. The figures have to be labeled, well explained and the results must be clearly discussed. When appropriate, it is recommended that you use the ***Hypothesis – Rationale – Experiments/data – Analysis – Results – Discussion/Conclusions – Limitation(s)*** framework to discuss your work.

The first sheet of the homework must certify that this is completely your work and list the students/people you have consulted or received help from (with your signature and date of submission). All online references used must be listed in the reference section at the end of the homework. It is sufficient to include Matlab code only in the online submission.

Good luck,
Barani Raman

Points for BME 572 students
Points for L41 5657 students

Problem 1. Implement the batch perceptron algorithm to obtain a linear discriminating function as described in Chapter 5 of Duda et al Pattern Classification book. Create linearly separable and non-linearly separable datasets with samples belonging to the two classes. Apply your perceptron algorithm to discriminate. Report your observation and analysis? Plot classification error vs. # of iterations, classification results, and the obtained decision boundary.

[20 pts]
[40 pts]

Problem 2: Using the same datasets used in problem 1, now create a linear classifier using Least Mean Squares (LMS) rule. Compare these results with the Perceptron algorithm results.

[20 pts]
[30 pts]

Problem 3. Generate 100 random (x, y) points, and run a 2D-lattice SOM with 100 neurons in a 10by10 lattice. Show that the SOM can perform density estimation by generating random points that belong to various distributions (uniform, Gaussian etc.).

[20 pts]
[30 pts]

For BME 572 students only

[40 pts]

Problem 4: Using back-propagation algorithm train a multilayer perceptron for the problem of recognizing handwritten digits. A popular dataset ('mnist_all.dat') comprising of training and testing samples of the different digits is provided in the homework folder. Each sample is 28x28 gray scale 8-bit image.

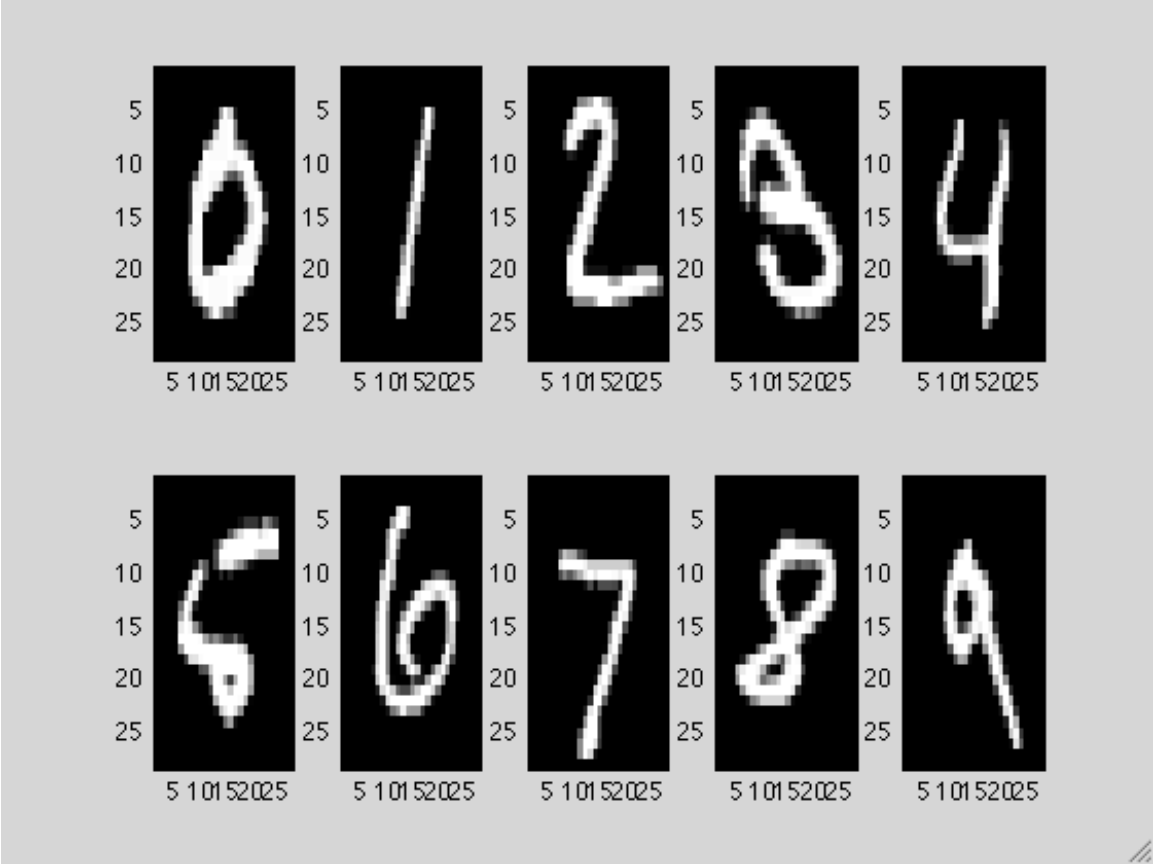

Figure 1: Sample of the nine handwritten digits in the MNIST dataset.

Training:
The Matrix train0 has the training samples for digit '0'. Each row has 784 columns corresponding to the 28x28 pixel (you can use reshape command to plot the digits; e.g. imagesc(reshape(test0(1,:),28,28)') plots first training sample for digit 0''.) Similarly, there is one dataset corresponding to each digit. You will train your network using the training samples only. You are free to choose a network of any size, and any non-linear activation function. Also, you are free to use any preprocessing technique or dimensionality reduction technique, or use only a subset of training samples, if you would like to reduce the complexity of the neural network or the training process.

Initialize the weight vectors to a very small random number between 0 and 0.1. This will help the network to converge better than equal weights or zero weights.

For non-linear activation two popular choices are the following:

Choice1: Logistic function

$$y_j(n) = \varphi_j(v_j(n)) = \frac{1}{1+\exp(-av_j(n))} \qquad a > 0 \quad -\infty < v_j(n) < \infty$$

$$\varphi_j'(v_j(n)) = a y_j(n)\left[1 - y_j(n)\right]$$

$$\Rightarrow \delta_j(n) = e_j(n)\varphi_j'(v_j(n)) = a\left[d_j(n) - o_j(n)\right]o_j(n)\left[1 - o_j(n)\right] \qquad j \rightarrow output\ node$$

$$\Rightarrow \delta_h(n) = \varphi_h'(v_h(n))\sum_j \delta_j(n) w_{jh}(n) = a y_h(n)\left[1 - y_h(n)\right]\sum_j \delta_j(n) w_{jh}(n) \qquad h \rightarrow hidden\ node$$

Choice2: Hyperbolic tangent function

$$y_j(n) = \varphi_j(v_j(n)) = a\tanh\left(bv_j(n)\right) \qquad a,b > 0$$

$$\varphi_j'(v_j(n)) = \frac{b}{a}\left[a - y_j(n)\right]\left[a + y_j(n)\right]$$

$$\Rightarrow \delta_j(n) = e_j(n)\varphi_j'(v_j(n)) = \frac{b}{a}\left[d_j(n) - o_j(n)\right]\left[a - o_j(n)\right]\left[a + o_j(n)\right] \qquad j \rightarrow output\ node$$

$$\Rightarrow \delta_h(n) = \varphi_h'(v_h(n))\sum_j \delta_j(n) w_{jh}(n) = \frac{b}{a}\left[a - y_h(n)\right]\left[a + y_h(n)\right]\sum_j \delta_j(n) w_{jh}(n) \qquad h \rightarrow hidden\ node$$

[Note: a, b are constants]

Testing:

The Matrix test0 has the test samples for digit '0'. Similarly, there is one corresponding to each digit. You will evaluate the performance of your network using the test samples only.

Show the evolution of the prediction error as a function of training iteration, final classification percentages for each digit, and the overall classification performance. Discuss your findings.

For reference about this popular dataset take a look at: http://yann.lecun.com/exdb/mnist/