Neurocomputing: Fundamentals of Computational Neuroscience

Assignment 2 due October 1 in class (15 points)

Team up with <u>one</u> other student of the class.

Write a program that implements a multiplayer mapping network (multilayer perceptron) that includes at least one hidden layer. Train this network on the pattern (letter) recognition task of assignment 1. The input data are in file http://www.cs.dal.ca/~tt/CSCI6508/assignment1/pattern1 (same as for tutorial 1 and assignment 1). As in assignment 1, each letter should be represented at the output layer with a single active output node, one for each of the 26 letters (local coding). Train the network using the error-backpropagation algorithm (see below for a summary of the learning algorithm).

Hint: this assignment is very similar to assignment 1, but you should use a network with a hidden layer. Specify your definition of an active node, that is, think about how you represent an active node in the output layer.

1. Specify all necessary details of your implementation (for example number of hidden nodes, gain function, additional post processing if this was necessary, …)

2. Plot the learning curve in which you show the performance of the network versus the training steps.

3. Evaluate the robustness of the network in recognizing noisy versions of the letter patterns. Plot a curve that shows the average recognition rate versus the noise level. The plot should include errorbars.

4. Compare the results to the results in assignment 1.

Summary of the error-backpropagation algorithm:

1. Initialize weights to small random values
2. Apply a sample pattern to the input nodes

   $$r_i^0 := r_i^{\mathrm{in}} = \xi_i^{\mathrm{in}}$$

3. Propagate input through the network by calculating the rates of nodes in successive layers l

   $$r_i^l = g(h_i^{l-1}) = g(\sum_j w_{ij}^l r_j^{l-1})$$

4. Compute the delta term for the output layer

   $$\delta_i^{\mathrm{out}} = g'(h_i^{\mathrm{out}})(\xi_i^{\mathrm{out}} - r_i^{\mathrm{out}})$$

5. Back-propagate delta terms through the network

   $$\delta_i^{l-1} = g'(h_i^{l-1}) \sum_j w_{ij}^l \delta_j^l$$

6. Update weight matrix by adding the term

   $$\Delta w_{ij}^l = k \delta_i^l r_j^{l-1}$$

7. Repeat steps 2–7 until error is sufficiently small