Neurocomputing: Fundamentals of Computational Neuroscience

Assignment 1 due September 24 in class (15 points)

Team up with one other student of the class.

Write a program that implements a mapping network (perceptron) with a single layer (not counting the input layer). The nodes should be binary threshold nodes. Train the network on the pattern (letter) recognition task of tutorial 1. The input data are in file http://www.cs.dal.ca/~tt/CSCI6508/assignment1/pattern1 (same as for tutorial 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 error-correction learning with an appropriate delta rule (Widrow-Hoff). See below for a summary of the learning algorithm.

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

2. 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.

3. Similarly, evaluate the robustness of your solution of the pattern recognition in tutorial 1. Discuss briefly the results of 2 and 3.

4. Discuss briefly the advantages and disadvantages of the two methods.

Summary of delta-learning algorithm:

Note: You should ignore the $g'(h)$ part in step 4 and calculate the delta term as difference between desired and actual output!

```
1. Initialize weights to random values
2. Apply a sample pattern to the input nodes
       $r_i^0 = r_i^{\text{in}} = \xi_i^{\text{in}}$
3. Calculate rate of the output nodes
       $r_i^{\text{out}} = g(\sum_j w_{ij} r_j^{\text{in}})$
4. Compute the delta term for the output layer
       $\delta_i = g'(h_i^{\text{out}})(\xi_i^{\text{out}} - r_i^{\text{out}})$
5. Update the weight matrix by adding the term
       $\Delta w_{ij} = k \delta_i r_j^{\text{in}}$
6. Repeat steps 2–5 until error is sufficiently small
```