CSci6903: Assignment Set # 2 (2 Pages)
Handed Out on: October 13, 1999
Due Date: October 27, 1999

Part 1: Problem 4.8, p. 125 of Mitchell's Text Book.

Part 2: Problem 5.4, p. 152 of Mitchell's Text Book.

Part 3: Training a Neural Network on a letter-recognition task. In this problem, you will be using tools from the "Matlab Neural Network toolbox" and "Matlab" in order to train and test a multi-layer perceptron, using Backpropagation (with Momentum) on a letter-recognition task. To help you get started, a Matlab Primer can be found at:

http://fourier.che.udel.edu/ cheg401/matlab.html.

Also, I own a book on Matlab that you can borrow (I only own one copy, however) and I have placed two files in /pub/nat/Assignment2: "mlp.m" and "data_for_mlp.m" that contain code very similar to the code you need to write for this assignment. To start matlab (once it is installed), simply type "matlab" at the unix prompt. A description of the data and the data itself can be found in /pub/nat/Assignment2 in the files called "letter-recognition.names" and "letter-recognition.data", respectively. Basically, the objective is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. In this problem, you are asked to compare four different learning schemes, all involving backpropagation:

Scheme 1: create a feedforward neural network containing 26 output units (one for each letter (i.e., class)).

Scheme 2: Repeat scheme 1, but rather than using 26 output units, use 5 of them and learn a binary encoding of the 26 classes (e.g., A corresponds to 00001; B corresponds to 00010; C corresponds to 00011, etc.).

Scheme 3: Translate the multi-class letter-recognition problem into a series of concept-learning problems (example: the letter "A" is associated with an output of 1; while every other letter is associated with an output of 0). Run backprop on these problems. Do not repeat this scheme with every letter in the alphabet, but repeat it with 2 different letters (in addition to letter "A")..

Scheme 4: Use an *autoassociator* to learn how to recognize one letter (again, run this experiment no more than three times with different letters). Define the "reconstruction error" as $\Sigma_{i=1}^{n}(x_i - o_i)^2$ where i is an index running through the number of input/output units; n is the total number of input/output units; x_i represents input # i; and o_i represents output # i (corresponding to x_i). Call an examplar "recognized" if its reconstruction error falls below a certain threshold (define this threshold yourself by simple inspection) and "non-recognized", otherwise.

Compare the four schemes in terms of *accuracy* and *efficiency* (# of epochs necessary for convergence to take place). Since there are many data points, divide your data set into two subsets (making sure that approximately every letter is equally represented in each set): Subset A and Subset B. In order to establish an appropriate number of hidden units, H, for each scheme as well as a stopping criterion, S, subdivide subset A into two equally distributed subsets again, Subset A.1 and Subset A.2, and use subset A.1 for training and Subset A.2 for validation with different values for H and S (**Notes:** (1) you can start by setting the learning rate, η, to 0.01 and momentum, α, to 0.9 and change them if that seems necessary. (2) Use a systematic method to select an optimal number of hidden units and maximum epoch number). Once all the parameters are established (η, α, H, and S) (please, note that the search for optimal parameters has to be repeated for each task), use subset B in a 5-fold cross-validation fashion and report (1) the error rates you obtained; (2) the optimal parameters you used for your experiments; (3) the 95% confidence interval for your cross-validation results; (4) the significance levels of each comparison (i.e., Scheme 1 versus Scheme 2, Scheme 1 versus Scheme 3, etc...).

Discuss your results, trying to explain why certain schemes were more accurate or efficient than others.

Comments:

- Please note that your networks will not converge unless you normalize the input data (i.e., project each attribute onto a [0, 1] range).
- Use the sigmoidal activation function in your hidden and output units in all cases.
- Part 3 of this assignment is likely to be very time consumming. You are thus encouraged to submit that part in groups of 2 where one person implements Schemes 1 and 3 while the other person implements Schemes 2 and 4. Submit Parts 1 and 2 individually and Part 3, in group. [Please talk to me if you would prefer doing this assignment alone or if you want to do it in a group of three: I will decrease or increase the experimental load accordingly.].
- This assignment will be time consumming (especially, Part 3). Start it early!