STA 414/2104, Spring 2007 — Assignment #3

Due at start of class on March 23. Note that this assignment is to be done by each student individually. You may discuss it in general terms with other students, but the work you hand in should be your own.

In this assignment you will try to recognize hand-written digits using an ensemble of multilayer perceptron networks trained using early stopping.

To help you along, I have provided R code for MLP training and early stopping that was part of a solution to an assignment for a previous course. It is for binary classification (not 10-way classification, as for digits), however, and doesn't form an ensemble. Note also that the notation used was different — the gamma parameters are weights for input to hidden unit connections, the beta parameters are weights for hidden to output unit connections, p is the number of inputs, q the number of hidden units, and M is the total number of parameters in the network. You will need to modify this code as appropriate for this assignment, or write your own version from scatch. You should use vector operations to operate on all training cases simultaneously, which greatly improves speed in R.

The data is from the MNIST dataset, which has 120,000 labelled digits. For this assignment, I have randomly selected a training set of 300 of these digits and a test set of 1000 digits. The image of each digit is a $28 \times 28 = 784$ pixel array, with pixel values being integers from 0 to 255. The label for each digit is an integer from 0 to 9. The data is available from the web page, as files `a3-x-train`, `a3-t-train`, `a3-x-test`, and `a3-t-test`. I have also supplied a little R function for plotting a digit.

You should model the conditional probability of the class given the pixel inputs for a case using a multilayer perceptron network with one layer of 8 hidden units with tanh activation function, and 10 output units (one for each class). The output unit values are obtained by exponentiating their summed inputs and then dividing by the sum over all 10 outputs, so that the output values can be interpreted as the probabilities of the 10 digit classes. See equation 5.25 in the text.

You should divide the 300 training cases into four sets of 75 (take consecutive blocks of 75 cases, so that this will be consistent between students, they are already in random order). You should train networks by gradient descent with early stopping four times, each time using one of the four sets of 75 as the validation set and the remaining 225 training cases as the estimation set. You should then make predictions for the 1000 test cases by averaging the class probabilities produced by the four networks you obtained in this way.

You should manually set the two learning rates (stepsizes for gradient descent, for connections into hidden units and into output units) and the total number of iterations, based on some initial exploration of what works reasonably well. You may also need to change the standard deviation of the distribution for initial weights. You should not just use the first values for these tuning parameters that seem to do OK, but instead do some exploration to see whether other values would work better.

You should use R functions to automatically find the iteration with smallest validation error within the total number you did, and to automatically average predictions from the four runs.

Once you think you are getting reasonable results, you should look at how good the predictions on test cases are, in terms of error rate. You should compare the error rate of the ensemble of four networks with the error rates obtained when using each of the four networks on their own. You can repeat the whole procedure for several selections of learning rates if you aren't sure what the best selections are, but you should note how may sets of learning rates you tried on the test data.

You should hand in a listing of your program (including the script you used to test it) with suitable but not excessive comments, the error rates you found, and a brief discussion of the results and of your experience with setting the learning rates by trial and error.