STA 414/2104, Spring 2012 — Assignment #2

Due at the start of class on March 8. Please hand it in on 8 1/2 by 11 inch paper, stapled in the upper left, with no other packaging.

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 particular, you should not leave any discussion with someone else with any written notes (either on paper or in electronic form).

In this assignment, you will modify the R functions for learning multilayer perceptron networks (with a binary response) that are on the course web page (the demo for week 6) in two ways. The first modification is to implement a variation on early stopping designed for situations in which some inputs are more relevant than others. The second modification is to add a general facility for training an ensemble of networks with early stopping, and then make predictions for test cases using this ensemble. You will then try using an ensemble of networks trained with the modified early stopping method on simple artificial data and data for a real application, and compare with an ensemble of networks trained with ordinary early stopping.

The modified early stopping method uses different learning rates for different weights in the network. With this method, the learning rate for weights on the connections from some input to the hidden units is larger if the sum of the squares of the derivatives with respect to the weights on the connections from this input is large. The hope is that weights on connections from inputs that are relevant will then be learned faster, and reach appropriate values before the weights on connections from irrelevant inputs become large enough to produce overfitting. If so, this will improve predictive performance, since early stopping will be able to select a point where most of the good stuff has been learned, but not much bad stuff has been learned.

This method is described in the following paper:

> Neal, R. M. (1998) "Assessing relevance determination methods using DELVE", in C. M. Bishop (editor), Neural Networks and Machine Learning, pp. 97-129, Springer-Verlag.

which is available from my web page at

> http://www.cs.utoronto.ca/∼radford/ard-delve.abstract.html

However, you don't have to read this paper to do this assignment, since the required details are below.

In the usual gradient ascent procedure, the weights, w, at some iteration are updated to weights $w' = w + \eta g$, where g is the gradient of the log likelihood with respect to the weights, and η is a learning rate, which needs to be chosen by trial and error. If we divide the weights into groups $w^{(1)}_{0j}$, $w^{(1)}_{kj}$, $w^{(2)}_0$, and $w^{(2)}_j$, for $k = 1, \ldots, p$ and $j = 1, \ldots, m$, and divide the gradient vector, g, in the same way, then in detail,

$$w^{(1)\prime}_{0j} = w^{(1)}_{0j} + \eta g^{(1)}_{0j}, \quad w^{(1)\prime}_{kj} = w^{(1)}_{kj} + \eta g^{(1)}_{kj}, \quad w^{(2)\prime}_0 = w^{(2)}_0 + \eta g^{(2)}_0, \quad w^{(2)\prime}_j = w^{(2)}_j + \eta g^{(2)}_j$$

In the modified method, this is changed to

$$w_{0j}^{(1)\prime} \;=\; w_{0j}^{(1)} + \eta g_{0j}^{(1)}, \quad w_{kj}^{(1)\prime} \;=\; w_{kj}^{(1)} + \eta_k g_{kj}^{(1)}, \quad w_0^{(2)\prime} \;=\; w_0^{(2)} + \eta g_0^{(2)}, \quad w_j^{(2)\prime} \;=\; w_j^{(2)} + \eta g_j^{(2)}$$

where the learning rate for connections out of input k is defined as

$$\eta_k \;=\; \eta\, G_k^4 \,/\, \max_{k'} G_{k'}^4, \qquad \text{with } G_k^2 \;=\; \sum_{j=1}^{m} [\, g_{kj}^{(1)} \,]^2$$

You should implement this procedure as part of the `mlp.train` function, adding another argument to `mlp.train`, called `rel.eta` that is TRUE to enable this method, and FALSE for the ordinary method. You will also need to add this as an argument to `mlp.cv`.

You should hand in a printout of these modified functions, which should be written in a readable style.

To implement ensembles of networks trained by early stopping, you should write a function `mlp.ensemble`, whose arguments are a vector of responses for training cases (`y`), a matrix of inputs for training cases (`X`), two learning rates (`eta1` and `eta2`) for $w^{(1)}$ and $w^{(2)}$, two penalty magnitudes (`lambda1` and `lambda2`), the `rel.eta` argument mentioned above, the number of parts (`S`) to split the training set into, the number of iterations to train for (`iters`), the number of hidden units in the network (`m`), a matrix of inputs for test cases (`Xtst`), and a `cv.plot` argument specifying whether plots should be produced. The value returned is a vector of predictions for test cases, in the form of the probability that the response is 1.

The `S` argument specifies how many parts the training set is divided into, with each of these `S` parts of the training set acting as the validation set for one training run. You may assume that the training cases are in random order, and hence divide the training set by just taking the first n/S cases, then the next n/S cases, etc. (adjusting suitably if n/S is not an integer). The predictions for test cases should be found by averaging the probability of the response being 1 based on the `S` sets of weights found with the `S` training runs.

You can see how to go about doing this in the script for the week 6 example, although there it is done in a non-general way that is specific to the data set used in that example.

You should hand in a printout of your new `mlp.ensemble` function, which should be written in a readable style.

You should test your ensemble learning function on two data sets provided on the course web site, both with the original early stopping method (specified with `rel.eta=FALSE`) and with the version intended to work better when some inputs are more relevant than others (specified with `rel.eta=TRUE`). You should evaluate predictive performance in terms of the log probability of the correct response (summing this over validation cases or test cases).

For these tests, you should standardize the inputs so they have mean zero and standard deviation one (as you did for one test in Assignment 1). You should set the `lambda2` argument to zero (so that the $w^{(2)}$ weights are not penalized) and try values of 0, 1, 2, 4, 8, and 16 for `lambda1`, selecting a value from among these by the average log probability of the response in

validation cases for the `S` training runs (at the iteration chosen by early stopping). You will have to adjust `eta1` and `eta2` by trial and error (quite small values may be necessary). You should choose the number of iterations so that the best iteration is unlikely to be later than the last one (at least in most runs), which may require several thousand iterations. For all tests, you should set `S` to 4, and use $m = 6$ hidden units.

The first data set has ten inputs. It was artificially generated using a simple function for the probability that the response is 1, which in fact depends on only the first of these inputs, so the other nine are irrelevant. A training set with 400 cases is provided (in random order), along with a test set with 2000 cases. You should train on the training set with both methods, using each of the values of `lambda1` specified above, and for each method select a value of `lambda1` based on the validation average log probability. You should then see how well the method performs on the test cases with that value of `lambda1`. (You may also want to see what the performance is with the other values of `lambda1` in order to see how well choosing `lambda1` based on validation performance works.)

The second data set is for a real application, but is also artificially generated. It is taken from the UCI repository of machine learning datasets, at

http://archive.ics.uci.edu/ml/datasets/MAGIC+Gamma+Telescope

The task is to distinguish between showers of photons caused by cosmic rays entering the upper atmosphere and similar showers caused by high-energy gamma rays. A simulation program is used to produce data thought to be typical of each kind of shower. The task is to learn how to distinguish the two kinds of showers using this simulated data. There are ten inputs, which may perhaps be of varying relevance to the task of predicting the response variable (which is 1 for gamma ray showers, and 0 for other showers).

I have provided two training sets or this problem, in random order, both with 1500 cases, along with a test set with 16020 cases. You should try out the two methods in the same manner as described above for the first data set, reporting the results when using each of the two training sets (on which training should be done completely separately).

You should hand in the R scripts you used to run these tests, the output produced, and your discussion of the results. The discussion should include an evaluation of whether the modified early stopping methods seems useful (on these data sets). You might find it helpful to look at the values of the $w^{(1)}$ weights found with the two methods. You should also discuss how good the validation performance is as a guide to test performance, and how much difference is seen between the two training sets for the second data set.