CS578 Fall 2007
Empirical Methods in Machine Learning and Data Mining
Homework Assignment #2
Due: Thursday, October 18, 2007 at 2:55pm *before* class

The goal of this assignment is to experiment with artificial neural nets trained with backpropagation, early stopping, and N-fold cross validation. For this assignment use the hw2.dta dataset available on the course web page. The goal is to predict the boolean variable in col 1 from the other 15 attributes. Each attribute has four values: a, c, t, g (DNA-like sequences). You will have to recode the data so that it can be given as input to a neural net. One way to do this is to replace each of the 15 attributes with four boolean attributes only one of which is on. Another way is to replace each attribute with a single attribute with four values (e.g., 1, 2, 3, 4). You will experiment with both codings.

You may implement backprop yourself, or use a commercial/public domain implementation. Note that it probably will take more time to install, learn to use, and modify someone else's implementation than to program backprop yourself, so we encourage you to code backprop yourself. In fact, implementing bp yourself counts as extra credit. If you decide to use someone else's package, it is up to you to install it and make sure that it will support the experiments needed for this assignment. One public domain package you might want to consider that runs on a variety of platforms is SNNS: Stuttgart Neural Network Simulator. There also is a Matlab toolbox for neural nets that is supposed to be pretty good. WEKA also includes a neural net simulator.

In this assignment we will use Accuracy, RMSE, and ROC Area as the performance measures. C-code for calculating ROC Area is available on the course web page. The program is called perf.c, and calculates a number of performance measures such as Accuracy, RMSE, and ROC Area.

EXPERIMENTS:

0: Code each attribute by splitting attributes into four boolean attributes. Make sure that the output is scaled 0-1 so that you can use a sigmoid output unit.

   Draw one random sample of 10,000 points from the data set to use for your experiments. Save the remaining 10,000 points as a final, final test set.

1: For neural nets, you need train sets (backprop sets), early stopping sets (technically part of the train set), and test sets. Use N-fold CV from the 10,000 cases for the train/test sets. The early stopping set should be held out of the train set. One way to do this is to split the data into 5 2k folds. Do backprop on folds 1-3 (60% of the train data), use fold 4 for early stopping (20% of the train data), and test on fold 5. Repeat this process 5 times for 5-fold CV. There are other ways to do this and you don't have to use 5-fold CV. Carefully explain how you choose to do N-fold CV. A diagram or table showing the splits might be helpful.

2: Train fully-connected feedforward neural nets using vanilla backpropagation with momentum. Every backprop implementation defines learning rate and momentum somewhat differently, and the definitions often vary when using batch mode (updating once per epoch -- full pass through the training set) or when updating per pattern, so you'll have to experiment with the parameter settings to find values that work well with your code. You can use batch mode, per pattern, or per group of pattern updating. (HINT: If the nets are fully trained after less than 200 passes through the train set you're probably training too fast. If the nets are taking more than 10^6 passes through the train set you're probably training slower than necessary.) Feel free to use any number of hidden units that seems to work. I often use 16 or 32 hidden units when starting on a new problem.

3: Compute accuracy, RMSE, and ROC Area on the train, stopping, and test sets. Show graphs of performance vs number of epochs for the train and early stopping sets. The performances on the test sets should be reported at the early stopping point. Are the early stopping points for accuracy, RMSE, and ROC Area the same?

4: Try an alternate coding of the input attributes such as converting each attribute to a four valued input such as 1,2,3,4. Repeat the experiment above. How does performance with this alternate coding compare with the boolean coding?

5: Using the boolean coding, experiment with different numbers of hidden units. You might try 1,2,4,8,16,32,64,... or even 1,4,16,64,... What size net yields best generalization performance?

EXTRA CREDIT -- do one or more of the following:

- there are different ways to convert the inputs a,c,g,t to an ordering such as 1,2,3,4. one is a->1, c->2, g->3, t->4. try different possibilities to see which representation works best.
- N-fold CV leaves you with N or more trained neural nets. compare the average prediction of these nets with the performance of each of the nets alone to see which works better. use the final, final test set of 14k cases as the test set for this comparison.
- recode the inputs so that they are symmetric about 0 (i.e., go negative and positive). does this change performance?
- try nets with two or more hidden layers. do they perform better than nets with one hidden layer? are they harder to train?
- experiment with different size training set (generate a learning curve). how does the performance change with the size of the training set? does the optimal net size change?
- do a study of the effect of altering the learning rate and momentum on the generalization performance of the nets
- is there a difference in generalization performance and training time between batch mode updates and per pattern or per group of pattern updating?
- compare weight decay with early stopping. does one perform better than the other? is one easier to use than the other?
- do feature selection to find a subset of the features that seems to perform better than using all the features
- do a sensitivity analysis to figure out what inputs the trained nets use most. sensitivity analysis can be done by looking at derivatives of the output of the net with respect to the inputs, or by experimenting with injecting noise into the inputs one at a time
- implement vanilla backprop with momentum for fully-connected feedforward neural nets containing one hidden layer and trained with squared error

Hand in a brief (5 pages or less for the main assignment) summary of the results with enough documentation so that we can see what you did and how you did it. Do not write a paper. This is homework, not a class project. You'll probably want to use the neural net code for the class project, so effort spent now to write good code or become familiar with the package you use should pay off later. Attach a copy of any code you write to the report you hand in.

Have fun.