Recitation Supplement:
Creating a Neural Network for Classification SAS EM

December 2, 2002

Introduction

Neural networks are flexible nonlinear models that can be used for regression and classification tasks, in the same way that linear/logistic regression and tree models can. Accordingly, to explore neural network modeling in SAS EM, we will use the same data set `myraw.xls` that was used in earlier recitations introducing linear regression and tree models. Recall that this is a data set extracted from a larger data set maintained by a national veteran's organization on people who have made charitable donations to them. We will build a standard neural network model to predict whether someone would contribute to a new fundraising campaign (by modeling the probability of response). Neural networks can also be used for regression tasks (e.g. predicting the amount contributed by a donor), but the extension is straightforward, so we will not discuss this here.

Procedure

The first few steps will be essentially the same as in the October 7 and October 21 recitations, providing a brief review of target profiling and imputation for missing data. The later steps will focus in more detail on the use of the Neural Network node.

1. Download and save `myraw.xls` from the course website:
   (`http://www.orie.cornell.edu/~davidr/or474`, under Course Data Sets, labeled as Donations data).

   Import the data into SAS, start Enterprise Miner, and create a new project. Maximize the EM window. This will make it easier to view the charts and diagrams that we will create.

2. Drag an Input Data Source node onto the diagram, open it, and input the data set. (Also, change the metadata sample to be the full data set.)

   Make the usual changes to the default assignments under the Variables tab:

   (a) Set the Model Role of TARGET_B to target. Set the Model Role of TARGET_D to rejected.

   (b) Set the Model Role of variables PETS and PCOWNERS to input and their Measurement type to binary.

3. Go to the target profile window (right-click on TARGET_B, choose Edit target profile...), and create a new profile (Edit ▷ Create New Profile). Make the new profile active (right-click on its cell in the Use column, select Set to use).

After making sure that the new profile is highlighted, go to the Assessment Information tab. Create a new profit matrix (right-click in the left-hand list and select Add) and make it active (highlight it, right-click on it, and choose Set to use). We will use the same profit/cost information as before: Enter 12.32 under column 1, row 1, for the net profit when we correctly predict a response, and -0.68 in column 1, row 0, for the mailing cost incurred when we incorrectly predict a response. Put 0's in both rows of column 0.

Select the Prior tab. Create a new prior vector (right-click in the left-hand list and choose Add) and make it active (highlight it, right-click on it, and choose Set to use). Enter the true population proportions in the right-hand table (0.05 for responders, 0.95 for non-responders).

Close the target profiles window and close the node (saving changes in both).

4. Connect a Data Partition node after the Input Data Source node and open it. Under the Partition tab, reset the Percentages for the data partitioning so that the Train data set gets 70%, the Validation set gets 30%, and the Test set gets 0%. (This choice is somewhat arbitrary, but it provides more training data needed by a flexible model like a neural network, while still allowing enough data for validation.)

   Close this node (choosing to save changes).

   Just as transforming variables can sometimes improve a standard logistic regression, it can be a useful precursor to neural network modeling. (Recall that logistic regression can, in fact, be viewed as a very simple neural network.) However, it is not quite as important for more complex neural network models, since these are already capable of approximating certain types of nonlinear relationships. Transformations will not be used in this analysis, but if they were desired, a Transform Variables node could be added at this point. (If this analysis were a regression problem rather than a classification problem, transformation of the target variable might be necessary for a good model fit if the error variance depends on the target value, especially when using the usual squared-error loss.)

5. Neural network models, like linear/logistic regression models (and unlike tree models), generally require imputation of missing predictor variables. Connect a Replacement node after the Data Partition node and open it. Look at the Imputation Methods subtab (under the Defaults tab). Change the default imputation method for both the interval and class variables to tree imputation.

   Now make the usual modifications for the three ambiguous variables: Under the Class Variables tab, right click on the cell in the HOMEOWNR row and Imputation Method column. Choose Select Method... ▷ set value... on the pop-up menus. Enter U in the box and click OK. Now do the same for PETS and PCOWNERS.

   Close this node (choosing to save changes).

6. Connect a Neural Network node (from the EM toolbar) after the Replacement node, and open it. You will see the same Data, Variables, Output, and Notes tabs that you

have seen before when opening a Regression or Tree node. The General tab and Basic (or Advanced) tab allow you to specify particular options for building neural networks.

Go to the General tab. By default, the criterion of Profit / Loss will be used to select the final model, which is suitable for the current analysis. The Advanced user interface box, if checked, makes the Advanced tab available instead of the Basic tab. The Advanced tab allows more control over the configuration of the neural network and may therefore be preferred by experienced users. The current analysis will be a simple one, so this box should be left unchecked. The remaining two options may be used to analyze the training process (the fitting of the neural network to the data). We will also leave these at the defaults. (See the help files for more information.)

Go to the Basic tab. This tab allows simple options for configuring the network. Click on the drop button beside the Network architecture text box. This will open a new window. The text box labeled Hidden neurons allows you to control how many units are in the hidden layer. You can specify this qualitatively (High, Moderate, or Low noise, or Noiseless data), in which case the number of hidden units is determined based on the number of inputs and the sample size and indicated in the greyed-out box on the right side. Alternatively, you can specify the number of hidden units directly with the Set number ... option. Choose this option, and set the number of hidden neurons to 3.

The Direct connections text box allows you to specify whether or not to include *direct connections* between input and output units, i.e. model terms that are linear in the input variables (before application of the output activation function). Leave this at the default setting of No to fit the usual type of neural network.

The Network architecture text box allows some limited options for defining the structure of the network. By default, this is Multilayer perceptron, the standard type of neural network. With this option, there will be a single hidden layer with logistic (tanh) activation functions, by default. (Because we have one binary target variable, there will naturally be one output unit with a logistic (softmax) activation function, by default.) Click and hold the drop menu button to see the other options, which include a generalized linear model (no hidden layer) and various types of radial basis function networks. Leave this option at its default (Multilayer perceptron).

Click OK to return to the main level of the Basic tab. The training of a neural network involves the optimization of some type of objective function (often the likelihood function) starting from a given set of initial *weights* (the parameters of a neural network model). The Preliminary runs text box allows you to specify the number of preliminary optimizations that are performed (usually based on a subset of data to speed up training) to find good starting values for the network weights, and thereby possibly avoid convergence to a suboptimal solution. The Training technique text box allows you to choose which optimization method to use. Because training can take excessive amounts of time for large data sets with many variables, a maximum training time can be set in the Runtime limit text box. Leave all of these at their default settings, for now.

Close the Neural Network node, choosing to save changes, and giving the model an appropriate name.

7. Run the Neural Network node. A Process Monitor window may momentarily appear, illustrating the progress of the training by showing the objective function value at each stage for the training and validation data. (If the program seems to freeze when this window appears, try clicking the Continue button.) When the training finishes, choose to view the results.

   The Tables tab is open by default. This simply gives tabular representations of several output data sets generated by the Neural Network node. You can access various tables using the drop menu at the top of the tab. The Fit statistics table is displayed by default.

   Click on the Model tab. Its General subtab gives the usual information. The Network subtab gives more detailed information about the model options chosen. This can be useful if you which to know which options were chosen by default, and the resulting configuration of the network, but interpreting the information requires some understanding of what the options mean.

   Click on the Weights tab. The table displayed here (Table subtab) gives the fitted parameter values for the neural network (one for each connection, plus appropriate bias terms). These parameters are identified by the connections with which they are associated, as defined by the node labels in the From and To columns. Input nodes are identified by the corresponding variable name (e.g. AGE), hidden nodes are labeled with a layer number and an index (e.g. H12 for the second node in the first hidden layer), and the output node is labeled with the output variable name. Bias terms are simply labeled as BIAS in the From column, and can be thought of as corresponding to a constant input of 1. Note that categorical variables are converted to numerical indicator variables with appropriately modified names, just as in linear/logistic regression. Click on the Graph subtab to see a graphical representation of this information. (Note that you may need to scroll to see all of the variables: choose the scroll tool or right-click on the graph.)

   Click on the Plot tab. The graphs here show the performance of the model on the training and validation data sets as a function of training (optimization) iteration number. The criterion being graphed appears to be Average Error, by default. You can change this by right-clicking on the graph and selecting a different option from the pop-up menu. The vertical line indicates which iteration was chosen to produce the optimal weights. Interestingly, it appears that the weights chosen are not those corresponding to the final result of the optimization, but rather those corresponding to the iteration for which the model selection criterion is greatest on the validation data. (To verify this in the current case, right-click on the graph and choose Profit to see graphs of the average profit as a function of iteration.) This is an example of model selection by *early stopping*; see Sec. 11.5.2 in the text. (By default, *weight decay* is not performed, and it is only possible to choose weight decay manually through an appropriate selection from the node's Advanced tab.)

   The remaining tabs (Code, Log, Output, and Notes) work as usual, and are less important in casual use. (There is no apparent way to view the connection diagram from the results browser, but such a diagram can be created in the Advanced tab when the node is opened, to allow interactive training.)

Try It Yourself

1. Add an Assessment node to the graph, connecting it after the Neural Network node. View different assessment charts for the model. Add a Regression node (with stepwise model selection based on profit for the validation data) in parallel with the Neural Network node. How do the two models compare on the basis of the assessment charts over the validation data?

2. Add a few more Neural Network nodes (in parallel with the original node) with different numbers of hidden units. Can you find a neural network model that outperforms logistic regression on this data set?

Created by Trevor Park on December 1, 2002