SESUG Paper 101-2019

Macros for Neural Network Modeling

Ross Bettinger, Silver Spring, MD

Abstract

We have written a suite of SAS® macros designed to facilitate the development of artificial neural network models. These macros perform tasks in the development sequence of data preprocessing, modeling, assessing, and scoring. In the modeling phase, PROC NLP is used to find the optimum set of weights for a multilayer perceptron network architecture. The neural network that results from this process may be stored as Base SAS data step code for scoring new data. These macros may be used for tutorial purposes in revealing the background details of implementing multilayer perceptron neural networks.

Keywords

Assessment, Base SAS, Enterprise Miner™, macro, modeling, multilayer perceptron, neural network, optimization, preprocessing, PROC NLP, SEMMA methodology, scoring

Introduction

Artificial neural networks are computer-based algorithms that are essential tools for pattern recognition in machine learning applications. The popularity of artificial neural networks is due to their ability to "learn" characteristic features of data and capture the relationships embedded in a collection of data as a set of weighted basis functions that span the set of representative features. One architecture, the multilayer perceptron (MLP), is based in concept on the networks of neurons and their connections found in brain tissue.

We have written several SAS macros that prepare data for neural net modeling, drive the learning process of extracting patterns from data, assess the accuracy of a candidate neural net model, and apply a neural net model to data to produce scores for later use.

Description of Macros

The SEMMA[1] methodology is frequently used to guide the efforts of data miners in building models. We select a sample of data to use in building a model, perform exploratory data analysis on the sample to familiarize ourselves with the variables and their contents, modify the variables' contents as appropriate, build candidate models, and assess the models' performance. If we are not satisfied with our results, we return to a previous stage of the work, make appropriate changes, and repeat our efforts in a forward manner. When we have become satisfied that our model adequately serves our purposes, we move on to the final stage: deploying the model and scoring new data.

We assume for this discussion that the first two phases of the SEMMA methodology have been completed, *viz.*

1. The population dataset has been sampled appropriately for the problem.

[1] "SEMMA" is the acronym for sequential steps in SAS Institute's data mining methodology. It stands for Sample, Explore, Modify, Model, Assess and it is meant to guide the efforts of data miners in building models.

a. The data have no missing values.[2]
b. The data have been cleaned as appropriate to the measurement scale of each type of variable.[3]
c. For a classification model, the event/nonevent proportions have been determined, and the data have been selected so as to represent those proportions.

2. Exploratory data analysis has been performed so that the analyst is familiar with the contents of each variable and with other combinations of variables as appropriate.

A complete set of options for each macro is contained in Appendix B.

Preprocessing

Preprocessing the data to be used for modeling may be necessary to improve the relationship between the target variable and individual predictors. This action may be necessary if there are outliers or extreme values in certain variables that distort the hypothetical functional relationship between target variable and predictor.

There are many transformations available to a modeler, based upon the measurement scale of the variable. For example, to correct for the distorting effects of outliers on continuous variables, a log transformation may be helpful. Also, standardizing a continuous variable[4] may be useful in creating a predictor that is more efficiently represented in the context of the problem to be solved than in its original format.[5]

The `%ANN_PREPROC` macro has features to perform preprocessing on a set of input data and produce a set of output data which contains variables corresponding to user-specified transformations. The macro header is shown below with default values, and examples of use will be given in the sequel.

```
%ANN_PREPROC( DSNIN  /* SAS dataset containing input and target vars       */
            , DSNOUT /* SAS dataset containing transformed variables       */
            , TARGET    /* target variable                                 */
            , BINARY=   /* list of binary-valued   vars in set { 0, 1 }    */
            , INTERVAL= /* list of interval-scaled vars                    */
            , NOMINAL=  /* list of nominal-scaled  vars                    */
            , ORDINAL=  /* list of ordinal-scaled  vars                    */
            , BIN_TRANS=dummy       /* encoding   for binary-valued   vars */
            , INT_TRANS=min_max     /* xformation for interval-scaled vars */
            , NOM_TRANS=glm         /* encoding   for nominal-scaled  vars */
            , ORD_TRANS=thermometer /* encoding   for ordinal-scaled  vars */
            , MISSING=              /* treatment of missing values         */
            )
```

[2] Missing values represent placeholders for values in observations that are unavailable or unknown. Many statistical and machine learning algorithms, including neural networks, cannot use observations with missing values in calculations, so either the missing value must be imputed or the entire observation must be discarded.

[3] See Appendix A for definitions of measurement scale.

[4] An interval variable may be standardized by subtracting its mean from each value and dividing by the standard deviation of the variable (*e.g.,* z-score), or it may be mapped into the interval (0, 1).

[5] For example, a variable with a wide range may, if unstandardized, saturate a neuron that uses a logistic combination function by forcing the backpropagation mechanism to assign extreme values close to 1. If the variable were standardized to (0, 1), then its mapping would be unchanged but its effect would be more in concert with other variables in the set of predictors.

Transformations are prescribed according to measurement scale, as listed in Table 1.

Table 1: Transformations Based on Measurement Scale

Variable Measurement Scale	Transformation
Binary	Dummy, Effect
Nominal	GLM
Ordinal	Thermometer
Interval	Arctan, min_max, sigmoid, tanh, z_score

Modeling

We represent artificial neural networks as connected weighted sums of input variables. In a feedforward MLP neural network, there are at least three layers: input, hidden[6], and output. Each layer feeds forward the combined weighted sums of input variables of the current layer to the next layer. Multilayer perceptrons were developed to expand the capabilities of single-layer perceptrons.

A single-layer neural network, called a *perceptron*, produces an output that is a linear combination of its inputs. Hence, it can separate data into disjoint classes only if the data are linearly separable. The input-output relationship for the two-class exclusive-or problem is

$$y = bias + w_1 x_1 + w_2 x_2$$

which is the equation of a straight line. It fails to solve the exclusive-or problem, in which the data points are not linearly separable. Figure 1 depicts a single-layer perceptron.

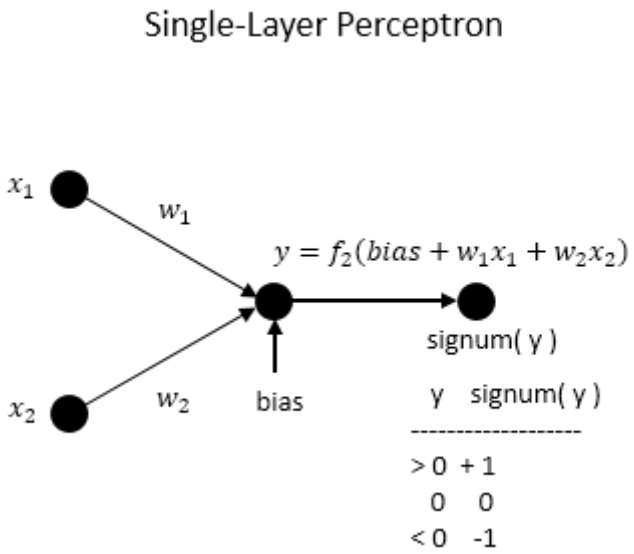

Figure 1: Single-Layer Perceptron

[6] There is at least one hidden layer; there may be several hidden layers, depending on the network architecture.

Figure 2 demonstrates that linear separation is not possible for the exclusive-or problem.

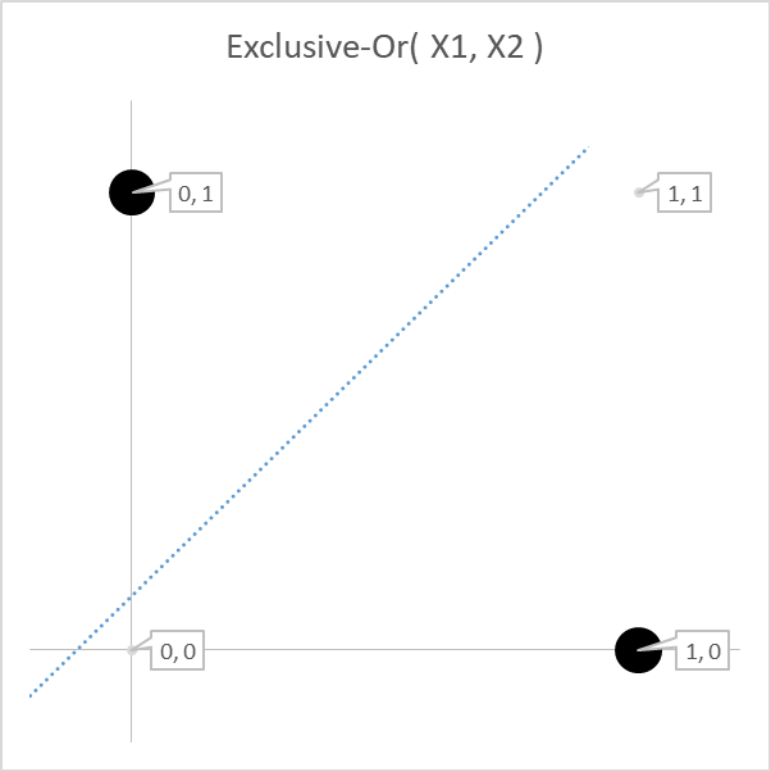

Figure 2: Exclusive-Or Problem

Later research on single-layer perceptrons led to multilayer perceptrons in which there is a set of mapping functions between the inputs and the outputs. The mapping functions are called nodes. The breakthrough advance was due to the insertion of a hidden layer of nodes between inputs and outputs. The hidden layer is represented by one or more *combination* functions that reduce one or more inputs into a single output. The output of one or more combinational nodes then becomes the input to an activation node. The output node is similarly represented by one or more *activation* functions that "squash" the combination function inputs into an appropriate output value or set of output values. There is a substantial body of knowledge available to inform the choice of combination and activation functions.[7] For the exclusive-or problem, the logistic function was used for combination and activation functions. Figure 3 represents a two-input, two-hidden node, single-output feedforward multilayer neural network.

Inputs x_1 and x_2 are combined by addition into inputs to activation function f_1 . The outputs from f_1 are scalar quantities h_1 and h_2 which are combined by addition into inputs to activation function f_2. The output from f_2 becomes the neural network output, y.

[7] See, e.g., https://www.kdnuggets.com/2016/08/role-activation-function-neural-network.html for a list of commonly-used functions, and https://en.wikipedia.org/wiki/Activation_function for an extensive list.

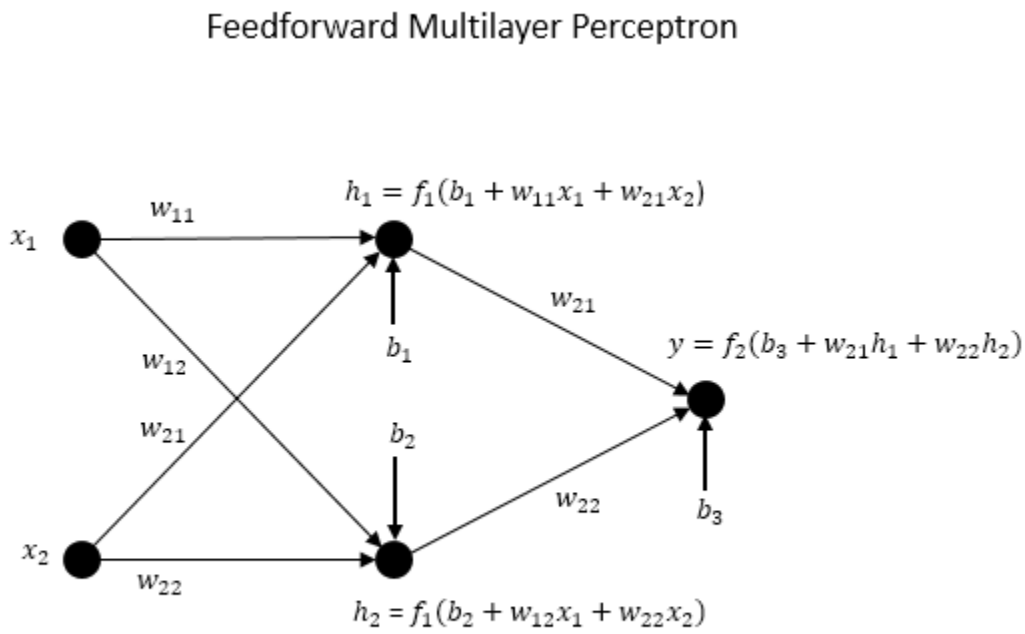

Figure 3: Two-Input, Two-Hidden, Single-Output MLP

The values of the input variables are fixed and the weights must be computed to represent the mapping between inputs and output(s) so as to optimize an objective function. Originally, backpropagation of errors[8] was used to update the weights, but it was computationally slow. More recent methods include nonlinear programming methods that are available in PROC NLP. Initial values for the network weights are set by the method specified in the `WGT_INIT` macro parameter[9].

The purpose of the `%ANN_NLP` macro is to 1) assemble programming statements for PROC NLP to use in representing a feedforward multilayer neural network and 2) invoke PROC NLP to solve for values of the network weights. The code fragment below demonstrates the use of the `%ANN_NLP` macro to solve for the network weights of the two-class exclusive-or problem.

```
%ANN_NLP( ANN.xor                   /* reference to input dataset       */
        , y_binary                  /* name of target variable          */
        , binary=x1 x2              /* names of binary variables        */
        , ACTIV_HIDDEN=tanh         /* activation fcn for hidden layer  */
        , ACTIV_OUT=logistic        /* activation fcn for output layer  */
        , DSNPREDICT=ANN.xor_scored /* reference to output dataset      */
        , HIDDEN=2                  /* # of nodes in hidden layer       */
        , grid_inc=100              /* # of grid points for PROC NLP    */
        , TECH=dbldog               /* use double dogleg algorithm      */
```

[8] Backpropagation of error was the original method used to update the weights of neural network nodes with respect to the error that each node produces. It is a gradient descent optimization technique designed to minimize the squared error of the activation function of the network [2].

[9] Table B.2 contains the algorithms available for initial network weight specification.

```
      , target_type=binary        /* measurement scale of target var  */
      , WGT_INIT=uniform          /* initialize network weights       */
      , MODELCODE=C:\_ann_.txt    /* reference to modeling code file  */
      )
```

The PROC NLP code produced by the %ANN_NLP macro and stored in file _ann_.txt is

```
proc nlp data=ANN.xor( keep= y_binary x1 x2 ) maxfunc=32000 maxiter=32000
outest=work.ann_parm_est out=ANN.xor_scored tech=dbldog best=0 ;
parms
b2_1=0,
w1_1_1=-0.010410964 to 0.0104109637 by 0.0002082193,
w1_2_1=-0.050461731 to 0.0504617309 by 0.0010092346,
b2_2=0,
w1_1_2=-0.53517399 to 0.5351739899 by 0.0107034798,
w1_2_2=-0.388470429 to 0.3884704286 by 0.0077694086,
b3_1=0,
w2_1_1=-0.368347776 to 0.3683477761 by 0.0073669555,
w2_2_1=-0.339574038 to 0.3395740381 by 0.0067914808,
;

/* begin neural network model code */

COMB2_1=b2_1+w1_1_1*x1+w1_2_1*x2 ;
H2_1=TANH( COMB2_1) ;
COMB2_2=b2_2+w1_1_2*x1+w1_2_2*x2 ;
H2_2=TANH( COMB2_2) ;
COMB3_1=b3_1+w2_1_1*H2_1+w2_2_1*H2_2 ;
H3_1=( COMB3_1) ;
p_y_binary = 0.5 *( tanh(H3_1 ) + 1 ) ;

/* end neural network model code */

loglik = ll_binary( y_binary, p_y_binary ) ;
max loglik ;
run ;
```

The tanh(x) function is used as the combination function for the hidden layer instead of the logistic(x) function because it has a wider linear range than the logistic(x) function. It can be used as the "squashing" function for the predicted value of the target variable, p_y_binary, because

$$logistic(x) = \frac{1}{2}(tanh(x) + 1)$$

The weight estimates computed by PROC NLP are

```
b2_1              0.254826
w1_1_1            0.376333
w1_2_1            0.558106
b2_2             -0.635517
w1_1_2            0.275550
w1_2_2            0.533867
b3_1           -124.727586
w2_1_1          172.481699
w2_2_1         -135.719723
```

where

- The neural network variables b_{ij} and w_{ijk} are represented in the PROC NLP code as bi_j and wi_j_k

- the prefix `b` represents *bias* and the subscripts `i,j` represent (current layer, node in current layer)
- the prefix `w` represents *weight* and the subscripts `i,j,k` represent (current layer, node in current layer, node in next layer)

The weight estimates are dependent on the initial values of the network parameters and will vary according to their initializations.

The MLP network equations after substitution of the bias and weight estimates are

$$h_1 = tanh(\ 0.254826 + 0.376333x_1 + 0.558106x_2\ )$$
$$h_2 = tanh(\ -0.635517 + 0.275550x_1 + 0.533867x_2\ )$$
$$h_3 = -124.727586 + 172.481699h_1 - 135.719723h_2$$
$$p\_y\_binary = logistic(\ h_3\ )$$

The MLP network equations produce the results shown in Table 2. We see that the estimates are very close to the actual values of the exclusive-or function.

Table 2: Results of XOR MLP

X1	X2	Y=XOR(X1, X2)	P_Y_Binary
0	0	0	0.0000184817
0	1	1	0.9999311295
1	0	1	1.0000000000
1	1	0	0.0000584415

Since the `%ANN_NLP` macro has a number of parameters and options, a more complete discussion is deferred to Appendix B.

Assessing

There is a provision in the `%ANN_NLP` macro to score the data used in building the network model. If the `&DSNPREDICT` macro parameter contains the name of a SAS dataset, PROC NLP will store the network predictions into the specified dataset for later use.

The `%ANN_ASSESS` macro uses the information stored in the `&DSNPREDICT` dataset to compute descriptive statistics and key performance indicators (KPI) derived from the actual data and predicted values of the data produced by the MLP model.

The performance metrics are specific to the type of model produced, e.g., classification or regression. If a binary classification model has been produced, the KPIs are lift, Brier score, ROC chart, area under the ROC curve, and 2x2 classification table. For a regression model, the KPIs are the R^2 statistic and the standard error of the regression.

The `%ANN_ASSESS` macro is invoked with the following parameters:

```
%ANN_ASSESS( DSNIN          /* dataset containing output from %ANN_NLP   */
           , TARGET         /* target variable                           */
           , TARGET_TYPE    /* target variable measurement scale         */
           , PRED_TARGET    /* predicted target variable                 */
           , N_QUANTILE=10  /* [optional] number of quantiles for lift   */
           , PLOTFILE=      /* [optional] file to which to send PDF plot */
           )
```

Scoring

The goal of model-building for classification and regression tasks is to capture the relationship between a target variable and its predictors in a well-defined set of functional statements. Once a model has been developed and has been validated, the next step is to use it to score observations and create predicted values of a target variable.

The %ANN_SCORE macro uses network parameter estimates produced by PROC NLP via the %ANN_NLP macro to apply the modeling statements stored in the &MODELCODE file to new data. An estimated value for the target variable is computed by passing the values of each observation's variables through the network. The macro is invoked with the following parameters:

```
%ANN_SCORE( DSNIN      /* SAS dataset of target var and input vars      */
          , DSNSCORE   /* SAS dataset of scored data                    */
          , DSNEST     /* SAS dataset of network parms and estimates    */
          , MODELCODE  /* name of modeling file produced by %ANN_NLP    */
          , SCORECODE= /* name of Base SAS program output by %ANN_SCORE */
          )
```

It produces values of &TARGETVAR in the dataset &DSNSCORE and writes the Base SAS representation of the neural network created by %ANN_NLP into file &SCORECODE. Since this file contains the representation of the neural network as Base SAS data step code, it may be used independently to score new data.

Example of Use

We demonstrate the use of the macros by applying them to the Home Equity dataset DMLHMEQ in library C:\Program Files\SASHome\SASFoundation\9.4\dmine\sample. We will score dataset DMTHMEQ and compare the results to those created by Enterprise Miner as a comparison.

The Home Equity loan scoring dataset contains data collected by a financial services company on clients to whom a home equity line of credit was extended. The company wants to use geographic, demographic, and financial information to build a credit scoring model to predict likelihood of a client to default on a loan. A detailed description is available in [3].

There is one binary target and 12 predictors. The variables are described in Table 3.

Table 3: Home Equity Loan Variables

Name	Role	Measurement Scale	Description
BAD	Target	Binary	A value of 1 indicates that the client defaulted on the loan or is seriously delinquent. A value of 0 indicates that the client paid off the loan.
CLAGE	Input	Interval	Age of the oldest credit line, measured in months
CLNO	Input	Interval	Number of credit lines
DEBTINC	Input	Interval	Debt-to-income ratio
DELINQ	Input	Interval	Number of delinquent credit lines
DEROG	Input	Interval	Number of major derogatory reports
JOB	Input	Nominal	Occupational category
LOAN	Input	Interval	Amount requested for the loan
MORTDUE	Input	Interval	Amount due on the existing mortgage
NINQ	Input	Interval	Number of recent credit inquiries
REASON	Input	Nominal	The value 'DebtCon' indicates that the loan was intended for debt consolidation. The value 'HomeImp' indicates that the loan was for home improvement.
VALUE	Input	Interval	Value of the current property
YOJ	Input	Interval	Years at the applicant's current job

Preprocessing consists of identifying the target variable, indicating measurement scales for each variable, indicating transformations for each variable, and specifying the treatment of missing values. The `%ANN_PREPROC` macro will read observations from dataset DMSAMPLE.DMLHMEQ and write to dataset WORK.HMEQ_LEARN all of the original variables in DMSAMPLE.DMLHMEQ and the preprocessed variables specified in the `nominal`, `ordinal`, and `interval` macro parameter lists. The preprocessed variables have 'X_' prefixed to their names to distinguish them from the original variables. Interval variables will be transformed into their z-score equivalents. Observations with missing values will be excluded from the output dataset.

Preprocessing

The `%ANN_PREPROC` macro invocation explicitly names the target and all predictors and specifies transformations according to measurement scale.

```
libname DMSAMPLE "C:\Program Files\SASHome\SASFoundation\9.4\dmine\sample" ;

%ANN_PREPROC( DMSAMPLE.dmlhmeq, WORK.hmeq_learn, bad
, binary   =bad
, nominal  =job reason
, ordinal  =
, interval =clage clno debtinc delinq derog loan mortdue ninq value yoj
, int_trans=z_score
, missing=drop
)
```

It is important to scale predictors so as to maintain homogeneity of magnitude. For example, if there is wide variety of housing values in the home loan data, homes with mortgages in the $100,000 range may be included with homes whose mortgages are in the $10,000,000 range. In this case, it is possible that

the error estimates produced by the computational algorithm will "saturate" the combination function chosen by the modeler. For the logistic combination function, inputs in the range $(-\infty, +\infty)$ produce outputs in the range (0, 1). If large positive inputs to the combination function are produced in the course of computing the network weights, then the network's weights may be driven close to 1. In this case, the weight update equations will fail to adjust the weights so as to minimize the errors in estimation because the gradient of the error function is too small to produce changes in output. A log transformation of the `mortdue` variable will transform the values into the range 5 to 7 and reduce the possibility of saturating the neuron's combination function.

Modeling

The `%ANN_NLP` macro invocation explicitly names the target and all predictors using the `%ANN_PREPROC` nomenclature.

```
%ANN_NLP( work.hmeq_learn, x_bad
, binary=
, nominal= x_job x_reason
, ordinal=
, interval=x_clage x_clno x_debtinc x_delinq x_derog x_loan x_mortdue x_ninq
           x_value x_yoj
, activ_hidden=tanh
, activ_out=tanh
, DSNEST=work.ann_parm_est
, DSNPREDICT=ANN.hmeq_learn_scored
, hidden=4
, ranseed=
, grid_inc=5
, tech=quanew
, target_type=binary
, wgt_init=uniform
, preproc=y
, modelcode=C:\Users\UserID\Documents\My SAS Files\Ann\_ann_hmeq.txt
)
```

Assessing

The `%ANN_ASSESS` macro uses the SAS dataset produced by the `%ANN_NLP` macro.

```
%ANN_ASSESS( ANN.hmeq_learn_scored, x_bad
, binary
, p_x_bad
, plotfile=C:\Users\UserID\Documents\My SAS Files\ANN\ann_assess_learn.pdf
)
```

Scoring

The `%ANN_SCORE` macro uses the Base SAS code in the `&MODELCODE` file produced by SAS dataset produced by the `%ANN_NLP` macro to score new observations.

```
%ANN_SCORE( work.hmeq_learn
, hmeq_learn_scored
, work.ann_parm_est
, C:\Users\UserID\Documents\My SAS Files\ANN\_ann_hmeq.txt
, scorecode=C:\Users\UserID\Documents\My SAS
Files\ANN\_ann_hmeq_learn_score.txt
)
```

```
/* score DMTHMEQ (Home Equity test) data */

%ANN_PREPROC( DMSAMPLE.dmthmeq, work.hmeq_test, bad
, binary=bad
, nominal=job reason
, ordinal=
, interval=clage clno debtinc delinq derog loan mortdue ninq value yoj
, int_trans=z_score
, missing=drop
)

%ANN_SCORE( work.hmeq_test
, work.hmeq_test_scored
, work.ann_parm_est
, C:\Users\UserID\Documents\My SAS Files\ANN\ConferencePaper\_ann_hmeq.txt
, scorecode=C:\Users\UserID\Documents\My SAS
Files\ANN\_ann_hmeq_test_score.txt
)

%ANN_ASSESS( work.hmeq_test_scored, x_bad
, binary
, p_x_bad
, plotfile=C:\Users\UserID\Documents\My SAS Files\ANN\ann_assess_test.pdf
)
```

Comparison of Results

We compared the model results generated by applying the %ANN_NLP macro to the results produced by SAS Enterprise Miner using the Home Equity data.

The Enterprise Miner model used the same network architecture and optimization algorithm as the %ANN_NLP macro. The Process Flow Diagram is shown below.

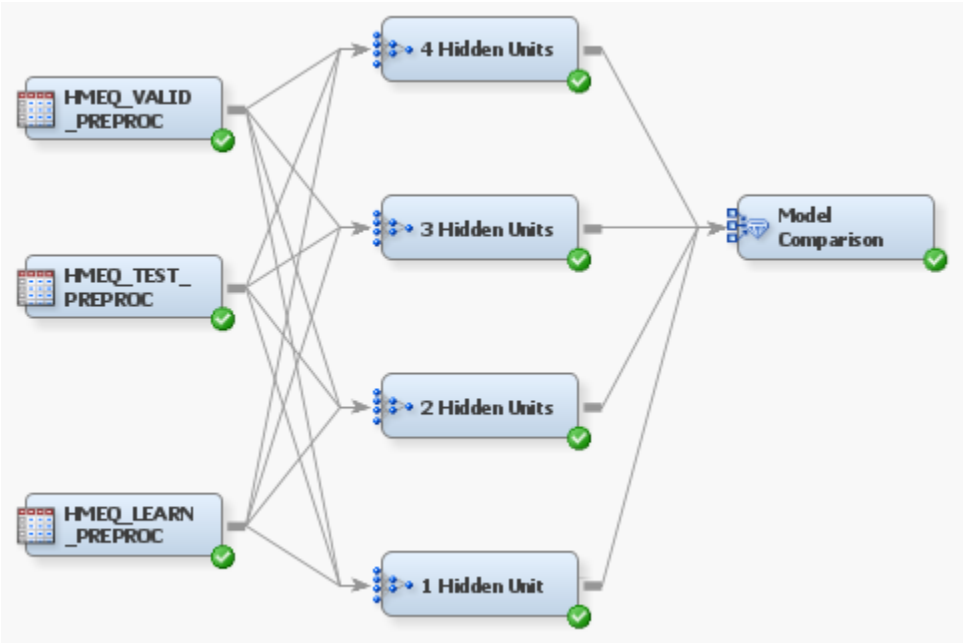

Figure 4: Enterprise Miner Process Flow Diagram

Table 4 shows the KPIs for the %ANN_NLP macro and Enterprise Miner.

Table 4: Comparing %ANN Macro KPIs to Enterprise Miner KPIs

# of Hidden Nodes	%ANN_NLP		Enterprise Miner	
	Top Decile Lift	Area Under ROC Curve	Top Decile Lift	Area Under ROC Curve
1	0.69	0.487	2.05	0.770
2	1.56	0.516	1.38	0.789
3	1.73	0.583	1.38	0.872
4	2.25	0.584	2.05	0.822

We see that the Enterprise Miner model outperforms the `%ANN_NLP` model in most cases. This result is expected because 1) the EM model used a validation dataset to provide feedback to the iterative weight determination process and thus avoid overfitting the model, and 2) the EM software has evolved over time through several generations of improvement by teams of skilled data scientists.

Summary

We have developed a set of SAS macros that preprocess data, generate SAS code for multilayer perceptron neural network models, assess the predictions of these models, and score new data. We demonstrated their use for a credit scoring problem and found that they produce similar results as the Enterprise Miner-developed model for the same data.

We propose that these macros have value for tutorial purposes in demonstrating how multilayer perceptron neural network models may be constructed and used without requiring Enterprise Miner software. These macros may be customized for specific applications and enhanced to include new features developed by researchers.

Appendix A

Reference [1] contains a description of the measurement scales commonly used in data mining.

- Nominal-scaled data are categorical in nature and do not have any natural ordering. There are typically a small number of distinct categories. Nominal data may be of character or numeric data type. There is no transitive relationship between two nominal data items. The distance between two nominal values is undefined.
- Ordinal-scaled data are categorical in nature and can be ordered relative to each other. There are typically a small number of distinct categories. Ordinal data may be of character or numeric data type. There is a transitive relationship between two ordinal data items. The distance between two ordinal values is undefined.
- Interval-scaled data are real-valued quantities. Interval-scaled data are ordered and there is a transitive relationship between two interval data items. The distance between two interval values is defined as their difference relative to an arbitrary zero point. The ratio between two interval-scaled data items is not defined.
- Ratio-scaled data are real-valued quantities. They extend interval-scaled data by defining the ratio between two ratio-scaled data items relative to a natural value of 0, which is lacking for interval-scaled data.

Appendix B

The %ANN_PREPROC macro header and parameter descriptions are given below.

```
%macro ANN_PREPROC( DSNIN  /* name of SAS dataset containing input and target variables */
                  , DSNOUT /* name of SAS dataset containing transformed variables      */
                  , TARGET /* target variable                                           */

                  /* inputs */

                  , BINARY=   /* binary-valued variables, numeric in set { 0, 1 }  */
                  , INTERVAL= /* interval-scaled variables                         */
                  , NOMINAL=  /* nominal-scaled variables                          */
                  , ORDINAL=  /* ordinal-scaled variables                          */

                  /* transformations */

                  , BIN_TRANS=dummy       /* [optional] encoding       for binary-valued variables   */
                  , INT_TRANS=min_max     /* [optional] transformation for interval-scaled variables */
                  , NOM_TRANS=glm         /* [optional] encoding       for nominal-scaled variables  */
                  , ORD_TRANS=thermometer /* [optional] encoding       for ordinal-scaled variables  */

                  /* data management */

                  , MISSING= /* [optional] treatment of missing values */
                  ) ;

   /* PURPOSE: create transformations on data prior to using it in %ANN for creating neural networks
    *
    * PARAMETERS:
    *  DSNIN     ::= name of SAS dataset containing input and target variables to use in building ANN
    *  DSNOUT    ::= name of SAS dataset containing transformed variables
    *  BINARY    ::= name(s) of binary-valued   variables
    *  INTERVAL  ::= name(s) of interval-scaled variables
    *  NOMINAL   ::= name(s) of nominal-scaled  variables
    *  ORDINAL   ::= name(s) of ordinal-scaled  variables
    *  BIN_TRANS ::= encoding       to be applied to binary-valued   variables (dummy)
    *  INT_TRANS ::= transformation to be applied to interval-scaled variables
    *                (arctan, min_max, sigmoid, tanh, z_score)
    *  NOM_TRANS ::= encoding       to be applied to nominal-scaled  variables (glm)
    *  ORD_TRANS ::= encoding       to be applied to ordinal-scaled  variables (thermometer)
```

```
*  MISSING   ::= [ keep | drop ] missing values
*
* NOTE:
*  if variable is followed by "/transform", e.g., INTERVAL= x1/arctan,
*  then arctan transform is applied to variable x1
*  this option is available for BINARY and INTERVAL-scaled variables ONLY
*
* NOMENCLATURE:
*  transformed variables are renamed using original name with "1" or "2" or ... or "n"
*  appended to name to signify transformed value
*
* EXAMPLE OF USE:
*
* %let BINARY    = YESNO M_F/effect                ;
* %let INTERVAL  = X1/arctan X2/min_max X3/sigmoid ; *** distinct transforms for each variable ***
* %let NOMINAL   = NOM1 NOM2                       ;
* %let ORDINAL   = ORD1 ORD2                       ;
* %let INT_TRANS = z_score                         ;
* %let NOM_TRANS = glm                             ;
* %let ORD_TRANS = thermometer                     ;
*
* %ANN_PREPROC( InputDataset, OutputDataset
*             , binary=&BINARY, interval=&INTERVAL, nominal=&NOMINAL, ordinal=&ORDINAL
*             , int_trans=&INT_TRANS, nom_trans=&NOM_TRANS, ord_trans=&ORD_TRANS
*             , missing=drop
*             )
*/
```

The %ANN_PREPROC macro has optional individual transformations according to the measurement scale of a variable. Table B.1 shows the possible transformations available as exemplified below.

Table B. 1: %ANN_PREPROC Transformations

Measurement Scale	Data Type	Transformation	Example	Effect
Nominal	Numeric Binary	Dummy encoding	BinaryVar/dummy	Maps BinaryVar into [0, 1]
	Numeric Binary	Effect encoding	BinaryVar/effect	Maps BinaryVar into [-1, 1]
	Character, Numeric	GLM encoding	NominalVar/GLM	Creates dummy var for each distinct value of NominalVar
Ordinal	Character, Numeric	Thermometer encoding	OrdinalVar/Thermometer	Creates dummy var for each value <= ordinal value
Interval	Numeric	Arctangent	IntervalVar/arctan	Maps z_score(IntervalVar) into (-π/2, π/2) radians
		Min_Max	IntervalVar/min_max	Maps IntervalVar into [-1, 1]
		Sigmoid	IntervalVar/sigmoid	Maps z_score(IntervalVar) into (0, 1)
		Hyperbolic Tangent	IntervalVar/tanh	Maps z_score(IntervalVar) into (-1, 1)
		Z-Score	IntervalVar/z_score	Maps IntervalVar into $\frac{(x-\mu)}{\sigma}$

The %ANN_NLP macro header and parameter descriptions are given below.

```
%macro ANN_NLP( DSNIN  /* SAS dataset containing input and target variables                                */
              , TARGET /* target variable                                                                  */

              /* inputs */

              , BINARY=   /* binary-valued   input variables                                               */
              , INTERVAL= /* interval-scaled input variables                                               */
              , NOMINAL=  /* nominal-scaled  input variables                                               */
              , ORDINAL=  /* ordinal-scaled  input variables                                               */
              , PREPROC=N /* flag to indicate that &DSNIN has variables preprocessed by %ANN_PREPROC       */

              /* ANN architecture */

              , ACTIV_HIDDEN=tanh  /* [optional] activation function(s) to use for hidden layer units(s) */
              , ACTIV_OUT=logistic /* [optional] activation function to use for output unit(s)           */
              , HIDDEN=2           /* [optional] list of number of hidden nodes per layer                */
              , RANSEED=           /* [optional] random number seed used to intialize biases and weights */
              , TARGET_ERR_FCN=    /* [optional] error function that is optimized to reduce network error*/
              , TARGET_TYPE=binary /* [optional] target type: BINARY, INTERVAL, NOMINAL, ORDINAL         */
              , WGT_INIT=          /* [optional] weight initialization probability density function      */

              /* solver options */

              , GRID_INC=5         /* [optional] grid increments for setting parameter values            */
              , ID=                /* [optional] name(s) of var(s) to use to identify obs for matching   */
              , MAXFUNC=32000      /* [optional] maximum # of function calls                             */
              , MAXITER=32000      /* [optional] maximum # of iterations in the optimization process     */
              , PRINTOPT=          /* [optional] print options                                           */
              , TECH=              /* [optional] default optimization technique                          */

              /* scoring options */

              , DSNPREDICT= /* [optional] dataset having predictions of neural network applied to &DSNIN */
              , DSNEST=     /* [optional] dataset containing all NN parameter estimates for reuse       */
              , MODELCODE=  /* [optional] output file to which to write model scoring code for reuse    */
              ) ;

    /* PURPOSE: develop artificial neural network using PROC NLP to solve for network weights
     *
     * PARAMETERS:
     *  DSNIN          ::= name of SAS dataset containing input and target variables to use in building ANN
     *  TARGET         ::= name of output variable whose values are to be predicted by ANN
```

```
*  INTERVAL       ::= name(s) of interval-scaled variables
*  NOMINAL        ::= name(s) of nominal-scaled variables (must be preprocessed)
*  ORDINAL        ::= name(s) of ordinal-scaled variables (must be preprocessed)
*  ACTIV_HIDDEN   ::= list of activation function(s) to be used for each unit in hidden layer
*  ACTIV_OUT      ::= activation function to be used for each unit in output layer
*  HIDDEN         ::= number of hidden layers
*  RANSEED        ::= random number seed used in weight initialization
*  TARGET_ERR_FCN ::= name of likelihood function that is optimized to produce neural network weights
*  TARGET_TYPE    ::= measurement scale of &TARGET, e.g., BINARY, NOMINAL, ORDINAL, INTERVAL
*  WGT_INIT       ::= probability dist from which to compute random weight initialization or a number
*  GRID_INC       ::= number of increments between end points of grid search interval
*  ID             ::= name(s) of variable(s) to use to identify obs for matching
*  MAXFUNC        ::= maximum # of function calls
*  MAXITER        ::= maximum # of iterations for &TECH optimization algorithm
*  TECH           ::= PROC NLP optimization technique
*  PRINTOPT       ::=
*  DSNPREDICT     ::= name of dataset to contain neural network predictions
*  MODELCODE      ::= name of file to which to write modeling code
*
* NOTE:    any preprocessed variables must be prefixed with "X_" so that appropriate programming
*          statements can be assembled to account for, e.g., preprocessed nominal or ordinal variables
*          that will require special treatment to represent multiple values. this situation will occur
*          if a nominal or ordinal variable has been preprocessed to be encoded with a GLM or a
*          thermometer encoding representation
*
*          a preprocessed nominal variable that contains multiple values will be represented as
*          n binary variables, one per distinct nominal value.
*          thus, if the variable Gender has the values Male, Female, Unknown, it must be preprocessed
*          into X_GENDER1, X_GENDER2, X_GENDER3
*          where each value (Male, Female, Unknown) will be represented as a binary variable using GLM
*          encoding
*
*          similarly, the ordinal variable Age that has the values Child, Teen, YoungAdult, Adult,
*          Senior must be preprocessed into X_AGE1, X_AGE2, X_AGE3, X_AGE4, X_AGE5
*          where each value (Child, Teen, ..., Senior) will be encoded using thermometer encoding
*
* ARCHITECTURE SCHEMA:
*  ANN architecture is input layer-hidden layer(s)-output layer
*  all node   indices refer to layer, input-to-[ hidden | output ] unit
*  all weight indices refer to layer, input-to-[ hidden | output ] unit, output-from-unit
*      Example: 2-2-1 ( 2 inputs, 1 hidden layer with 2 nodes, 1 output layer with 1 node
```

```
*      X11 -> H21
*      X11 -> H22
*      X12 -> H21
*      X12 -> H22
*      H21 -> O31
*      H22 -> O31
*
*      CombinationFunction21 = bias21 + w111 * X11 + w121 * X12
*      CombinationFunction22 = bias22 + w121 * X11 + w122 * X12
*
*      HiddenNode21 = ActivationFunction21( CombinationFunction21 )
*      HiddenNode22 = ActivationFunction22( CombinationFunction22 )
*
*      OutputNode31 = ActivationFunction31( bias31 + w131 * HiddenNode21 + w132 * HiddenNode22 )
*
* EXAMPLE OF USE:
*
* %let INTERVAL     = INPUT1 INPUT2 X_INPUT3 ; *** variable INPUT3 was preprocessed ***
* %let HIDDEN       = 1                      ;
* %let TARGET       = prob_event             ;
* %let ACTIV_HIDDEN = tanh                   ;
* %let ACTIV_OUT    = logistic               ;
*
* %ANN_NLP( InputDataset
*         , interval=&INTERVAL
*         , HIDDEN=&HIDDEN
*         , TARGET=&TARGET
*         , ACTIV_HIDDEN=&ACTIV_HIDDEN
*         , ACTIV_OUT=&ACTIV_OUT
*         )
*/
```

The %ANN_NLP macro has parameter options that may be specified according to a specific model. Table B.2 shows the options available.

Table B. 2: %ANN_NLP Parameter Options

Parameter	Meaning	Option	Value
ACTIV_HIDDEN	Hidden layer activation function	Exponential Linear Logistic Hyperbolic **T**angent	EXP LINEAR LOGISTIC TANH
ACTIV_OUT	Output layer activation function	Huber Logistic Normal Poisson Softmax	HUBER LOGISTIC NORMAL POISSON SOFTMAX
WGT_INIT	Initial network weight randomization distribution	He Normal Uniform	Sample from $N(0, \sqrt{\frac{2}{\#ofInputNodes}})$ Sample from $N(0, \sqrt{\frac{1}{\#ofInputNodes}})$ Sample from $U(\sqrt{\frac{1}{\#ofInputNodes}})$

The %ANN_ASSESS macro header and parameter descriptions are given below.

```
%macro ANN_ASSESS( DSNIN         /* SAS dataset having target variable and predicted value of target var */
                 , TARGET        /* target variable                                                      */
                 , TARGET_TYPE   /* target variable measurement scale                                    */
                 , PRED_TARGET   /* predicted target variable                                            */
                 , N_QUANTILE=10 /* [optional] # of quantiles into which to group obs for BINARY target  */
                 , PLOTFILE=     /* [optional] file to which to send PDF plot                            */
         ) ;

    /* PURPOSE: compute performance statistics to assess a model
     *
     * PARAMETERS:
     *  DSNIN      ::= name of SAS dataset having target variable and predicted value of target variable
     *  TARGET     ::= name of variable whose values are to be predicted
     *  PRED       ::= name of predicted value of &TARGET
     *  N_QUANTILE ::= number of groups to create for &TARGET_TYPE = BINARY
     *  PLOTFILE   ::= path/name of plot file to which to write graphical output
     *
     * EXAMPLE OF USE:
     *
     * %let DSN1       = TrainingDataset ;
     * %let TARGET     = target_var      ;
     * %let MEAS_SCALE = binary          ;
     * %let PRED       = prob_event      ;
     *
     * %ANN_ASSESS( DSN1, &TARGET, &MEAS_SCALE, &PRED )
     */
```

The %ANN_SCORE macro header and parameter descriptions are given below.

```
%macro ANN_SCORE( DSNIN      /* SAS dataset having target variable and input vars to be used in scoring */
                , DSNSCORE   /* SAS dataset containing scored data                                      */
                , DSNEST     /* SAS dataset containing network parameters and estimates                 */
                , MODELCODE  /* name of text file produced by %ANN that contains neural network model   */
                , SCORECODE= /* [optional] name of program produced from &MODELCODE having scoring code */
                ) ;

    /* PURPOSE: use previously-developed neural network parameter estimates to score data
     *
     * PARAMETERS:
     *  DSNIN     ::= name of SAS dataset containing independent variables used in building %ANN model
     *  DSNSCORE  ::= name of SAS dataset that will contain predicted target value produced by %ANN model
     *  DSNEST    ::= name of SAS dataset containing estimates for parameters used in %ANN model
     *  MODELCODE ::= name of text file produced by %ANN
     *  SCORECODE ::= name of text file produced by %ANN_SCORE that contains scoring code
     *
     * EXAMPLE OF USE:
     *
     * %let DSN_IN     = TrainingDataset       ;
     * %let DSN_SCORE  = ScoredData            ;
     * %let DSN_EST    = estimated_network_wts ;
     * %let MODEL_CODE = path/ANN_MODEL.txt    ;
     * %let SCORE_CODE = path/ANN_SCORE.txt    ;
     *
     * %ANN_SCORE( &DSN_IN, &DSN_OUT, &DSN_EST, &MODEL_CODE, SCORECODE=&SCORE_CODE )
     */
```

References

[1] Conover, W.J. (1971). *Practical Nonparametric Statistics*, John Wiley & Sons, Inc., New York.

[2] https://en.wikipedia.org/wiki/Backpropagation

[3] Data Mining Using SAS(R) Enterprise Miner(TM): A Case Study Approach, Third Edition, http://support.sas.com/documentation/cdl/en/emcs/66392/HTML/default/viewer.htm#n12yb479smsxxsn0zphu0n8a0nz2.htm

Bibliography

SAS Institute Inc. (2014). *SAS/OR® 13.2 User's Guide: Mathematical Programming Legacy Procedures,* Cary, NC: SAS Institute Inc.

Acknowledgements

We thank Bruce Gilsen and Joseph Naraguma for their generosity in taking the time to review this document.

Contact Information

Your comments and questions are valued and encouraged. The code is available upon request. Contact the author at:

Name:	Ross Bettinger
Enterprise:	Consultant
E-mail:	rsbettinger@gmail.com