Chapter 5: Neural Network Models to Predict Response and Risk

Introduction

- We will now turn to using SAS Enterprise Miner to derive and interpret neural network models.
- This chapter develops two neural network models using simulated data.
- The first model is a response model with a binary target.
- The second is a risk model with an ordinal target.

Target Variables for the Models

- The target variable for the response model is RESP, which takes the value of 0 if there is no response and the value of 1 if there is a response.
- The neural network produces a model to predict the probability of response based on the values of a set of input variables for a given customer.
- The output of the **Neural Network** node gives probabilities Pr(*RESP* = 0 | *X*) and Pr(*RESP* =1| *X*) , where *X* is a given set of inputs.

Target Variables for the Models

- The target variable for the risk model is a discrete version of *loss frequency,* defined as the number of losses or accidents per car-year, where car-year is defined as the duration of the insurance policy multiplied by the number of cars insured under the policy.
- If the policy has been in effect for four months and one car is covered
- under the policy, then the number of car-years is 4/12 at the end of the fourth month.
- *See details in our text (chapter 5).*

Neural Network Node Details

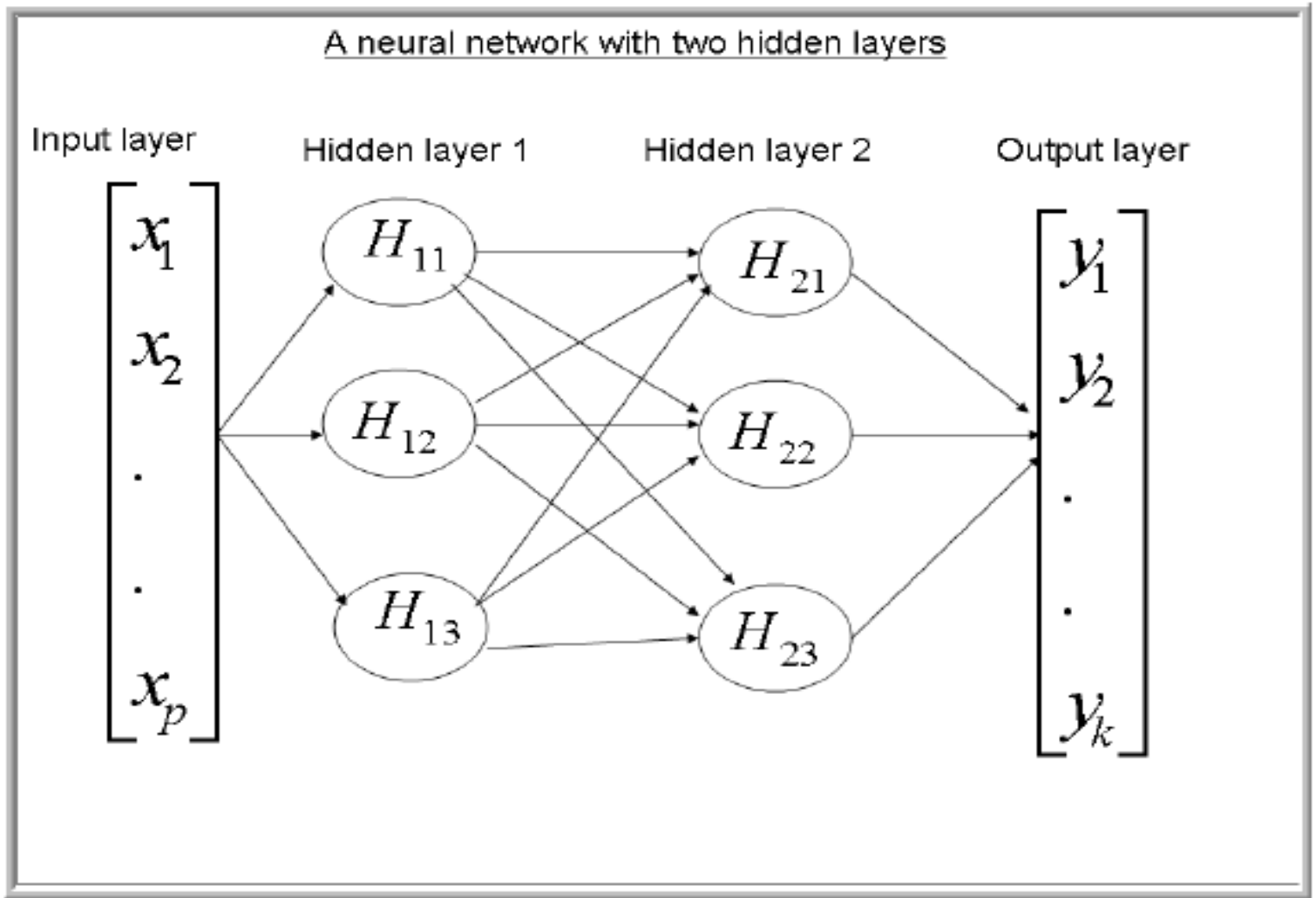
- Neural network models use mathematical functions to map inputs into outputs.
- When the target is categorical, the outputs of a neural network are the probabilities of the target levels.
- The neural network model for risk (in the example Chapter 5) gives formulas to calculate the probability of the variable LOSSFRQ taking each of the values 0, 1, or 2, whereas the neural network model for response provides formulas to calculate the probability of response and non-response.
- Both of these models use the complex nonlinear transformations available in the **Neural Network** node of SAS Enterprise Miner.

A General Example of a Neural Network Model

- To help you understand the models presented later, here is a general description of a neural network with two hidden layers1 and an output layer.
- Suppose a data set consists of *n* records, and each record contains, along with a person's response to a direct mail campaign, some measures of demographic, economic, and other characteristics of the person.
- These measures are often referred to as *explanatory variables* or *inputs*.
- *See Chapter 5 for details.*

A General Example of a Neural Network Model

Display 5.0A



Input Layer and Hidden Layers

- This layer passes the inputs to the next layer in the network either without transforming them or, in the case of interval-scaled inputs, after standardizing the inputs. Categorical inputs are converted to dummy variables.
- For each record, there are *p* inputs: $x_1, x_2, \ldots\ldots x_p$.
- There are three hidden units in each of the two hidden layers in this examp
- *For Details See Chapter 5.*

Activation Function of the Output Layer

- The *activation function* is a formula for calculating the target values from the inputs coming from the final hidden layer.
- The outputs of the final hidden layer are first combined as shown in Equation 5.13 (see Chapter 5, Section 5.2.4 for details).

Estimation of Weights in a Neural Network Model

The weights are estimated iteratively using the training data set in such a way that the error function specified by the user is minimized. In the case of a response model, we use the following Bernoulli error function:

$$E = -2\sum_{i=1}^{n}\left\{ y_i \ln\frac{\pi(W,X_i)}{y_i} + (1-y_i)\ln\frac{1-\pi(W,X_i)}{1-y_i}\right\} \qquad (5.16)$$

where π is the estimated probability of response. It is a function of the vector of weights, W, and the vector of explanatory variables for the i^{th} person, X_i. The variable y_i is the observed response of the i^{th} person, in which $y_i = 1$ if the i^{th} person responded to direct mail and $y_i = 0$ if the i^{th} person did not respond. If y_i=1 for the i^{th} observation, then its contribution to the error function (Equation 5.16) is $-2\log\pi(W,X_i)$; if $y_i = 0$, then the contribution is $-2\log(1-\pi(W,X_i))$.

A Neural Network Model to Predict Response

- This section discusses the neural network model developed to predict the response to a planned direct mail campaign.
- The campaign's purpose is to solicit customers for a hypothetical insurance company.
- A two-layered network with one hidden layer was chosen. Three units are included in the hidden layer. In the hidden layer, the combination function chosen is linear, and the activation function is hyperbolic tangent.
- In the output layer, a logistic activation function and Bernoulli error function are used.
- The logistic activation function results in a logistic regression type model with non-linear transformation of the inputs, as shown in Equation 5.14 in Section 5.2.4.

A Neural Network Model to Predict Loss Frequency in Auto Insurance

- The premium that an insurance company charges a customer is based on the degree of risk of monetary loss to which the customer exposes the insurance company. The higher the risk, the higher the premium the company charges.
- For proper rate setting, it is essential to predict the degree of risk associated with each current or prospective customer.
- Neural networks can be used to develop models to predict the risk associated with each individual.

Example

- We develop a neural network model to predict loss frequency at the customer level.
- We use the target variable LOSSFRQ that is a discrete version of loss frequency.
- The definition of LOSSFRQ is presented in the beginning of Chapter 5, Section 5.1.1.
- If the target is a discrete form of a continuous variable with more than

The goal of the model developed here is to estimate the conditional probabilities $\Pr(LOSSFRQ = 0 \mid X)$, $\Pr(LOSSFRQ = 1 \mid X)$, and $\Pr(LOSSFRQ = 2 \mid X)$, where X is a set of inputs or explanatory variables.

Loss Frequency as an Ordinal Target

- In order for the **Neural Network** node to treat the target variable LOSSFRQ as an ordinal variable, we should set its measurement level to Ordinal in the data source.
- See Chapter 5, Section 5.5.1.

Classification of Risks for Rate Setting in Auto Insurance with Predicted Probabilities

- The probabilities generated by the neural network model can be used to classify risk for two purposes:

• to select new customers

• to determine premiums to be charged according to the predicted risk

- For each record in the data set, you can compute the expected LOSSFRQ as:

$$E(lossfrq \mid X_i) = Pr(lossfrq = 0 \mid X_i) * 0 + Pr(lossfrq = 1 \mid X_i) * 1 + Pr(lossfrq = 2 \mid X_i) * 2 .$$

Customers can be ranked by this expected frequency and assigned to different risk groups.

Alternative Specifications of the Neural Networks

- The general Neural Network model presented in section 5.2 consists of linear combinations functions (Equations 5.1, 5.3, 5.5, 5.7, 5.9, and 5.11) and hyperbolic tangent activation functions

  (Equations 5.2, 5.4, 5.6, 5.8, 5.10, and 5.12) in the hidden layers, and a linear combination

  function (Equation 5.13) and a logistic activation function (Equation 5.14) in the output layer.
- Networks of the type presented in Equations 5.1-5.14 are called multilayer perceptrons and use linear combination functions and sigmoid activation functions in the hidden layers.
- Sigmoid activation functions are S-shaped, and they are shown in Displays 5.0B, 5.0C, 5.0D and 5.0E.
- You are also introduce to other types of networks known as **Radial Basis Function (RBF) networks**, which have different types of combination activation functions.
- You can build both Multilayer Perceptron (MLP) networks and RBF networks using the **Neural Network** node.

A Radial Basis Function (RBF) Neural Network

$$y_i = \sum_{k=1}^{M} w_k \phi_k (X_i) + w_0 \qquad (5.33)$$

where y_i is the target variable for i^{th} case (record or observation), w_k is the weight for the k^{th} basis function, w_0 is the bias, ϕ_k is the k^{th} basis function, $X_i = (x_{i1}, x_{i2}, \ldots\ldots, x_{ip})$ is a vector of inputs for the i^{th} case, M is the number of basis functions, p is the number of inputs, and N is the number of cases (observations) in the Training data set. The weight multiplied by the basis function plus the bias can be thought of as the output of a hidden unit in a single layer network.

An example of a basis function is:

$$\phi_k (X_i) = \exp\left(-\frac{\|X_i - \mu_k\|}{2\sigma_k} \right), \text{ where } X_i = (x_{i1}, x_{i2}, \ldots\ldots, x_{ip}),\ \mu_k = (\mu_{1k}, \mu_{2k}, \ldots\ldots, \mu_{pk}),$$

i stands for the i^{th} observation and k stands for the k^{th} basis function. $\|X_i - \mu_k\|$ is the Squared Euclidean distance between the vectors X_i and μ_k. μ_k and σ_k are the center and width of the k^{th} basis function. The values of the center and width are determined during the training process.

Comparison of Alternative Built-in Architectures of the Neural Network Node

- You can produce a number of neural network models by specifying alternative architectures in the **Neural Network** node.
- You can then make a selection from among these models, based on lift or other measures, using the **Model Comparison** node.
- We can compare the built-in architectures listed in the Display 5.46 (see your text).

Normalized Radial Basis Function with Equal Widths and Heights (NRBFEQ)

In this case, the combination function for the k^{th} hidden unit is $\eta_k = -b^2 \sum_{j=1}^{p} (w_{jk} - x_j)^2$, where w_{jk} are the weights iteratively calculated by the **Neural Network** node, x_j is the j^{th} input (standardized), and b is the square root of the reciprocal of the width. The height k^{th} hidden unit is = $a_k = 1$. Since $\log(1) = 0$, it does not appear in the combination function shown above. Here it is implied that $f = 1$ (See equation 5.36).

AutoNeural Node

- As its name suggests, the **AutoNeural** node automatically configures a Neural Network model.
- It uses default combination functions and error functions.
- The algorithm tests different activation functions and selects the one that is optimum.

DMNeural Node

- **DMNeural** node fits a non linear equation using bucketed principal components as inputs.
- The model derives the principal components from the inputs in the training data set.
- As explained in Chapter 2, the principal components are weighted sums of the original inputs, the weights being the eigenvectors of the variance covariance or correlation matrix of the inputs.
- Since each observation has a set of inputs, you can construct a Principal Component value for each observation from the inputs of that observation.
- The Principal components can be viewed as new variables constructed from the original inputs.

Dmine Regression Node

- The **Dmine Regression** node generates a Logistic Regression for a binary target.
- The estimation of the Logistic Regression proceeds in three steps.
- In the first step, a preliminary selection is made, based on Minimum R-Square. For the original variables, the Rsquare is calculated from a regression of the target on each input; for the binned variables, it is calculated from a one-way Analysis of Variance (ANOVA).

Dmine Regression Node

- In the second step, a sequential forward selection process is used. This process starts by selecting the input variable that has the highest correlation coefficient with the target. A regression equation (model) is estimated with the selected input. At each successive step of the sequence, an additional input variable that provides the largest incremental contribution to the Model R-Square is added to the regression. If the lower bound for the incremental contribution to the Model R-Square is reached, the selection process stops.
- In the third step, the algorithm estimates a logistic regression (in the case of a binary target) with a single input, namely the estimated value or prediction calculated in the first step.

Summary

- A neural network is essentially nothing more than a complex nonlinear function of the inputs. Dividing the network into different layers and different units within each layer makes it very flexible. A large number of nonlinear functions can be generated and fitted to the data by means of different architectural specifications.
- The combination and activation functions in the hidden layers and in the target layer are key elements of the *architecture* of a neural network.

Summary

- You specify the architecture by setting the **Architecture** property of the **Neural Network** node to User or to one of the built-in architecture specifications.
- When you set the **Architecture** property to User, you can specify different combinations of Hidden Layer Combination Functions, Hidden Layer Activation Functions, Target Layer Combination Functions, and Target Layer Activation Functions. These combinations produce a large number of potential neural network models.

Summary

If you want to use a built-in architecture, set the **Architecture** property to one of the following values:

GLM (generalized linear model, not discussed in this book),

MLP (multilayer perceptron),

ORBFEQ (Ordinary Radial Basis Function with Equal Widths and Heights),

ORBFUN (Ordinary Radial Basis Function with Unequal Widths),

NRBFEH (Normalized Radial Basis Function with Equal Heights and Unequal Widths),

NRBFEV (Normalized Radial Basis Function with Equal Volumes),

NRBFEW (Normalized Radial Basis Function with Equal Widths and Unequal Heights),

NRBFEQ (Normalized Radial Basis Function with Equal Widths and Heights, and NRBFUN(Normalized Radial Basis Functions with Unequal Widths and Heights).

Summary

• Each built-in architecture comes with a specific Hidden Layer Combination Function and a specific Hidden Layer Activation Function.

• While the specification of the Hidden Layer Combination and Activation functions can be based on such criteria as model fit and model generalization, the selection of the Target Layer Activation

Function should also be guided by theoretical considerations and the type of output you are interested in.

Summary

• In addition to the **Neural Network** node, three additional nodes, namely **DMNeural**, **AutoNeural**, and **Dmine Regression** nodes, were demonstrated.

However, there is no significant difference in predictive performance of the models developed by these three nodes in the example used in the text.

• The response and risk models developed here show you how to configure neural networks in such a way that they are consistent with economic and statistical theory, how to interpret the results correctly, and how to use the SAS data sets and tables created by the **Neural Network** node to generate

customized reports.