

# Chapter 5 Introduction to Predictive Modeling: Neural Networks and Other Modeling Tools

<b>0.1</b>	<b>Introduction.....</b>	<b>Error! Bookmark not defined.</b>
<b>0.2</b>	<b>A Section Title.....</b>	<b>Error! Bookmark not defined.</b>
	Demonstration: <Type title of demo here.>.....	<b>Error! Bookmark not defined.</b>
	Exercises .....	<b>Error! Bookmark not defined.</b>
<b>0.3</b>	<b>Chapter Summary.....</b>	<b>Error! Bookmark not defined.</b>
<b>0.4</b>	<b>Solutions .....</b>	<b>Error! Bookmark not defined.</b>
	Solutions to Exercises .....	<b>Error! Bookmark not defined.</b>
	Solutions to Student Activities (Polls/Quizzes) .....	<b>Error! Bookmark not defined.</b>



## 5.1 Introduction

### Model Essentials – Neural Networks

▶ Predict new cases.	Prediction formula
▶ Select useful inputs.	None
▶ Optimize complexity.	Stopped training

3 ...

With its exotic sounding name, a neural network model (formally, for the models discussed in this course, *multi-layer perceptrons*) is often regarded as a mysterious and powerful predictive tool. Perhaps surprisingly, the most typical form of the model is, in fact, a natural extension of a regression model.

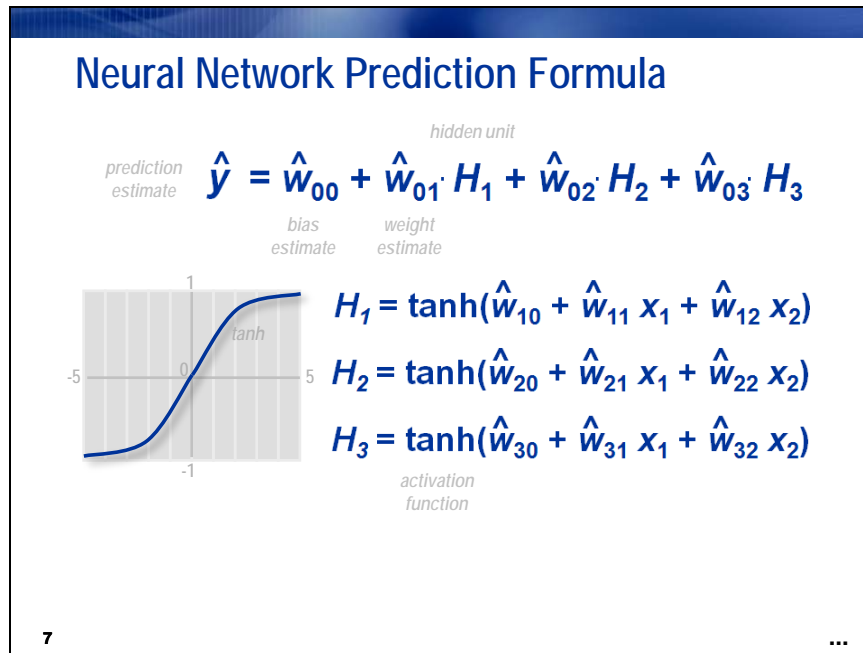
The prediction formula used to predict new cases is similar to a regression's, but with an interesting and flexible addition. This addition enables a properly trained neural network to model virtually any association between input and target variables. Flexibility comes at a price, however, because the problem of input selection is not easily addressed by a neural network. The inability to select inputs is offset (somewhat) by a complexity optimization algorithm named *stopped training*. Stopped training can reduce the chances of overfitting, even in the presence of redundant and irrelevant inputs.

## Model Essentials – Neural Networks

▶ Predict new cases.	Prediction formula
▶ Select useful inputs.	None
▶ Optimize complexity.	Stopped training

5 ...

Like regressions, neural networks predict cases using a mathematical equation involving the values of the input variables.



A neural network can be thought of as a regression model on a set of derived inputs, called *hidden units*. In turn, the hidden units can be thought of as regressions on the original inputs. The hidden unit “regressions” include a default link function (in neural network language, an *activation function*), the hyperbolic tangent. The hyperbolic tangent is a shift and rescaling of the logistic function introduced in Chapter 3.

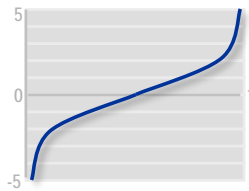
Because of a neural network’s biological roots, its components receive different names from corresponding components of a regression model. Instead of an intercept term, a neural network has a *bias* term. Instead of parameter estimates, a neural network has *weight estimates*.

What makes neural networks interesting is their ability to approximate virtually any continuous association between the inputs and the target. You simply specify the correct number of hidden units and find reasonable values for the weights. Specifying the correct number of hidden units involves some trial and error. Finding reasonable values for the weights is done by least squares estimation (for interval-valued targets).

## Neural Network Binary Prediction Formula

$$\log\left(\frac{\hat{p}}{1-\hat{p}}\right) = \hat{w}_{00} + \hat{w}_{01} H_1 + \hat{w}_{02} H_2 + \hat{w}_{03} H_3$$

logit  
link function



$$H_1 = \tanh(\hat{w}_{10} + \hat{w}_{11} x_1 + \hat{w}_{12} x_2)$$

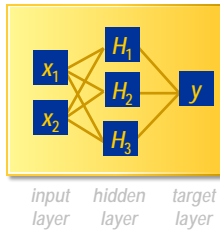
$$H_2 = \tanh(\hat{w}_{20} + \hat{w}_{21} x_1 + \hat{w}_{22} x_2)$$

$$H_3 = \tanh(\hat{w}_{30} + \hat{w}_{31} x_1 + \hat{w}_{32} x_2)$$

When the target variable is binary, as in the demonstration data, the main neural network regression equation receives the same logit link function featured in logistic regression. As with logistic regression, the weight estimation process changes from least squares to maximum likelihood.

## Neural Network Diagram

$$\log\left(\frac{\hat{p}}{1-\hat{p}}\right) = \hat{w}_{00} + \hat{w}_{01} H_1 + \hat{w}_{02} H_2 + \hat{w}_{03} H_3$$

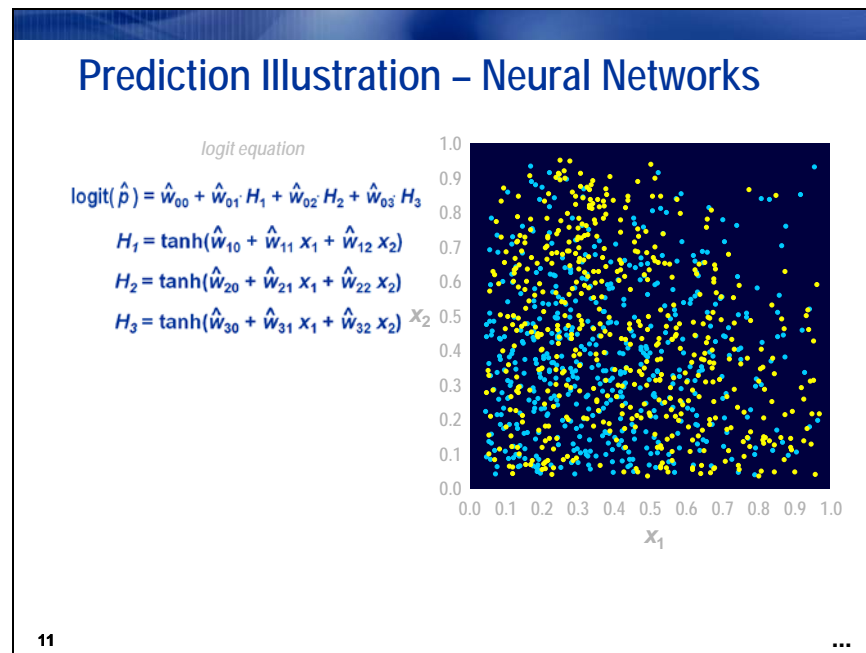


$$H_1 = \tanh(\hat{w}_{10} + \hat{w}_{11} x_1 + \hat{w}_{12} x_2)$$

$$H_2 = \tanh(\hat{w}_{20} + \hat{w}_{21} x_1 + \hat{w}_{22} x_2)$$

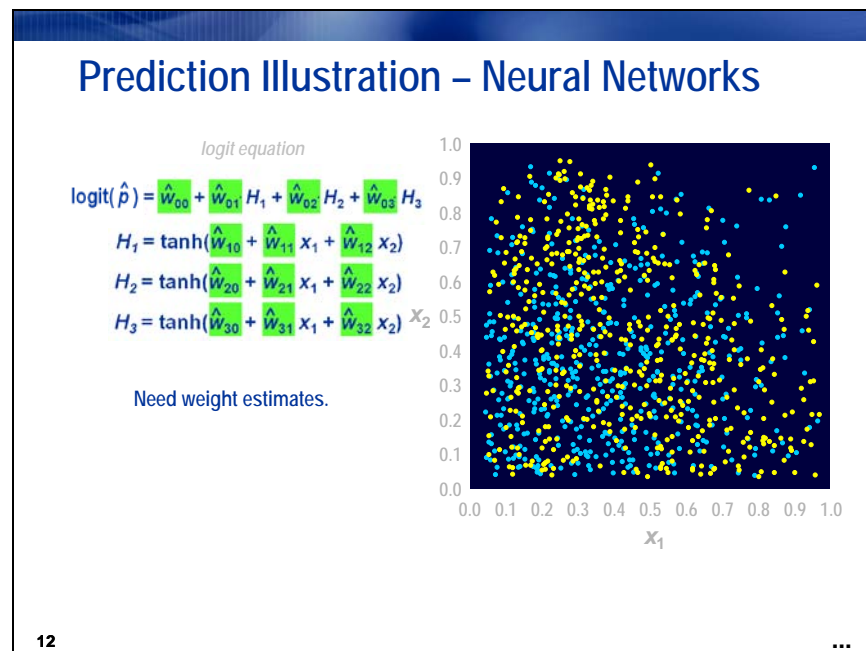
$$H_3 = \tanh(\hat{w}_{30} + \hat{w}_{31} x_1 + \hat{w}_{32} x_2)$$

Multi-layer perceptron models were originally inspired by neurophysiology and the interconnections between neurons, and they are often represented by a network diagram instead of an equation. The basic model form arranges neurons in layers. The first layer, called the *input layer*, connects to a layer of neurons called a *hidden layer*, which, in turn, connects to a final layer called the *target* or *output layer*. Each element in the diagram has a counterpart in the network equation. The blocks in the diagram correspond to inputs, hidden units, and target variables. The block interconnections correspond to the network equation weights.



To demonstrate the properties of a neural network model, consider again the two-color prediction problem. As always, the goal is to predict the target color based on the location in the unit square.

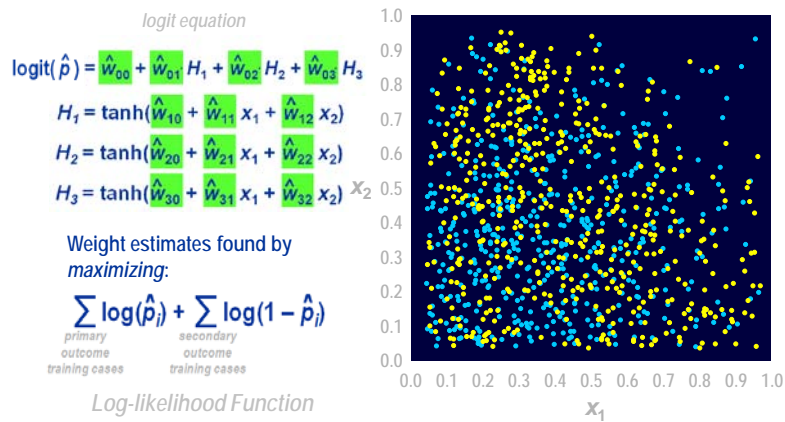
As with regressions, the predictions can be decisions, rankings, or estimates. The logit equation produces a ranking or logit score.



As with a regression model, the primary task is to obtain parameter or, in the neural network case, weight estimates that result in accurate predictions. A major difference between a neural network and a regression model, however, is the number of values to be estimated and the complicated relationship between the weights.



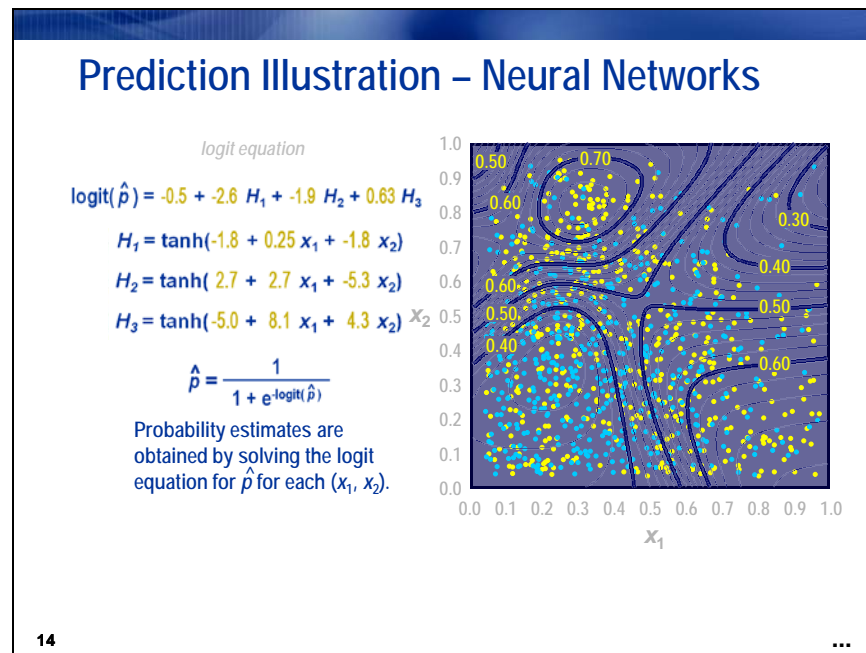
## Prediction Illustration – Neural Networks



13

...

For a binary target, the weight estimation process is driven by an attempt to maximize the log-likelihood function. Unfortunately, in the case of a neural network model, the maximization process is complicated by local optima as well as a tendency to create overfit models. A technique illustrated in the next section (usually) overcomes these difficulties and produces a reasonable model.



Even a relatively simple neural network with three hidden units permits elaborate associations between the inputs and the target. While the model might be simple, explanation of the model is decidedly not. This lack of explicability is frequently cited as a major disadvantage of a neural network. Of course, complex input/target associations are difficult to explain no matter what technique is used to model them. Neural networks should not be faulted, assuming that they correctly modeled this association.

After the prediction formula is established, obtaining a prediction is strictly a matter of plugging the input measurements into the hidden unit expressions. In the same way as with regression models, you can obtain a prediction estimate using the logistic function.

## Neural Nets: Beyond the Prediction Formula

- ▶ Manage missing values.
- ▶ Handle extreme or unusual values.
- ▶ Use non-numeric inputs.
- ▶ Account for nonlinearities.
- ▶ Interpret the model.

16

Neural networks share some of the same prediction complications with their regression kin.

The most prevalent problem is missing values. Like regressions, neural networks require a complete record for estimation and scoring. Neural networks resolve this complication in the same way that regression does, by imputation.

Extreme or unusual values also present a problem for neural networks. The problem is mitigated somewhat by the hyperbolic tangent activation functions in the hidden units. These functions squash extreme input values between -1 and +1.

Nonnumeric inputs pose less of a complication to a properly tuned neural network than they do to regressions. This is mainly due to the complexity optimization process described in the next section.

Unlike standard regression models, neural networks easily accommodate nonlinear and nonadditive associations between inputs and target. In fact, the main challenge is over-accommodation, that is, falsely discovering nonlinearities and interactions.

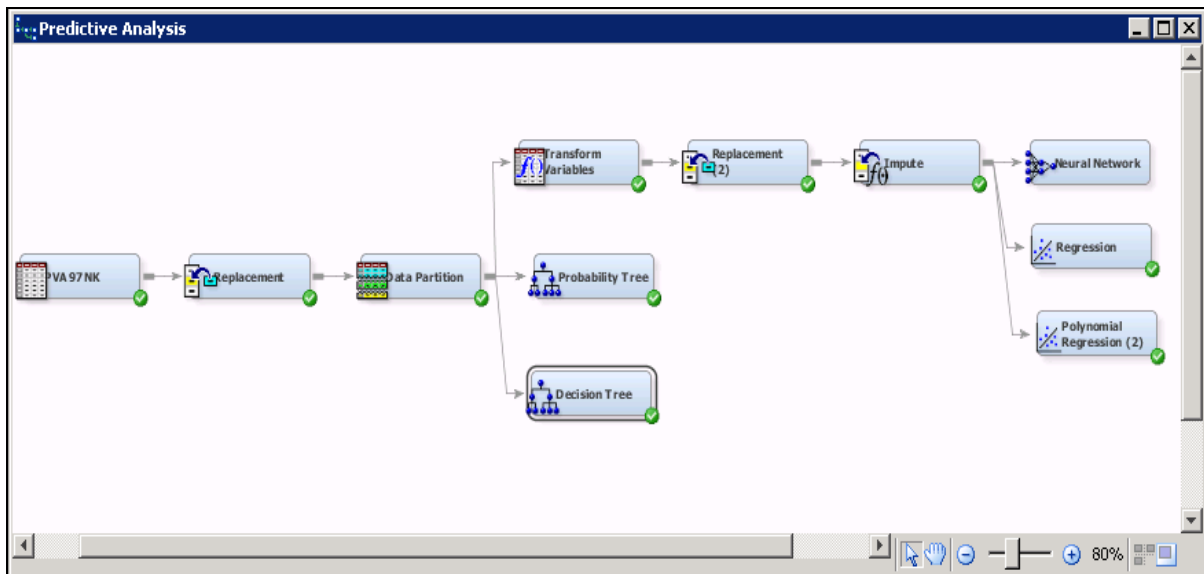
Interpreting a neural network model can be difficult, and while it is an issue, it is one with no easy solution.



## Training a Neural Network


Several tools in SAS Enterprise Miner include the term *neural* in their name. The Neural Network tool is the most useful of these. (The AutoNeural and DM Neural tools are described later in the chapter.)

1. Select the **Model** tab.
2. Drag a **Neural Network** tool into the diagram workspace.
3. Connect the **Impute** node to the **Neural Network** node.



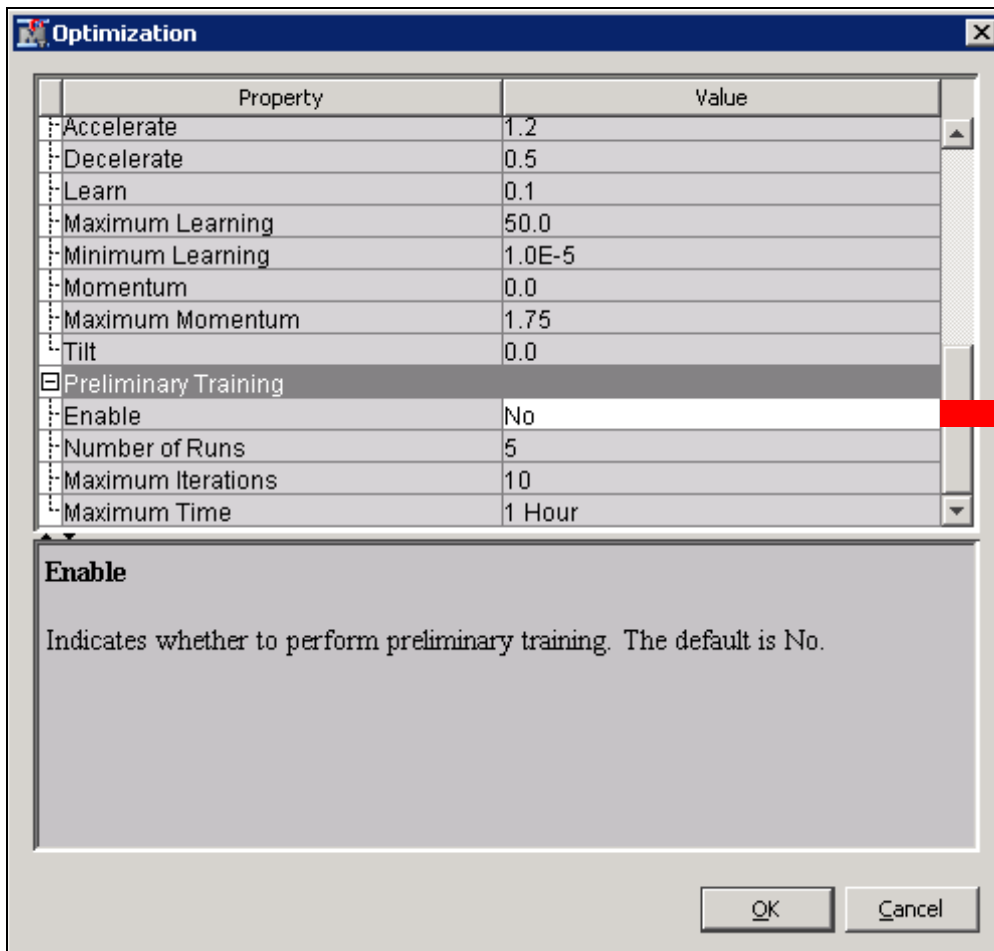
With the diagram configured as shown, the Neural Network node takes advantage of the transformations, replacements, and imputations prepared for the Regression node.

The neural network has a default option for so-called “preliminary training.”

4. Select **Optimization** ⇨  from the Neural Network Properties panel.

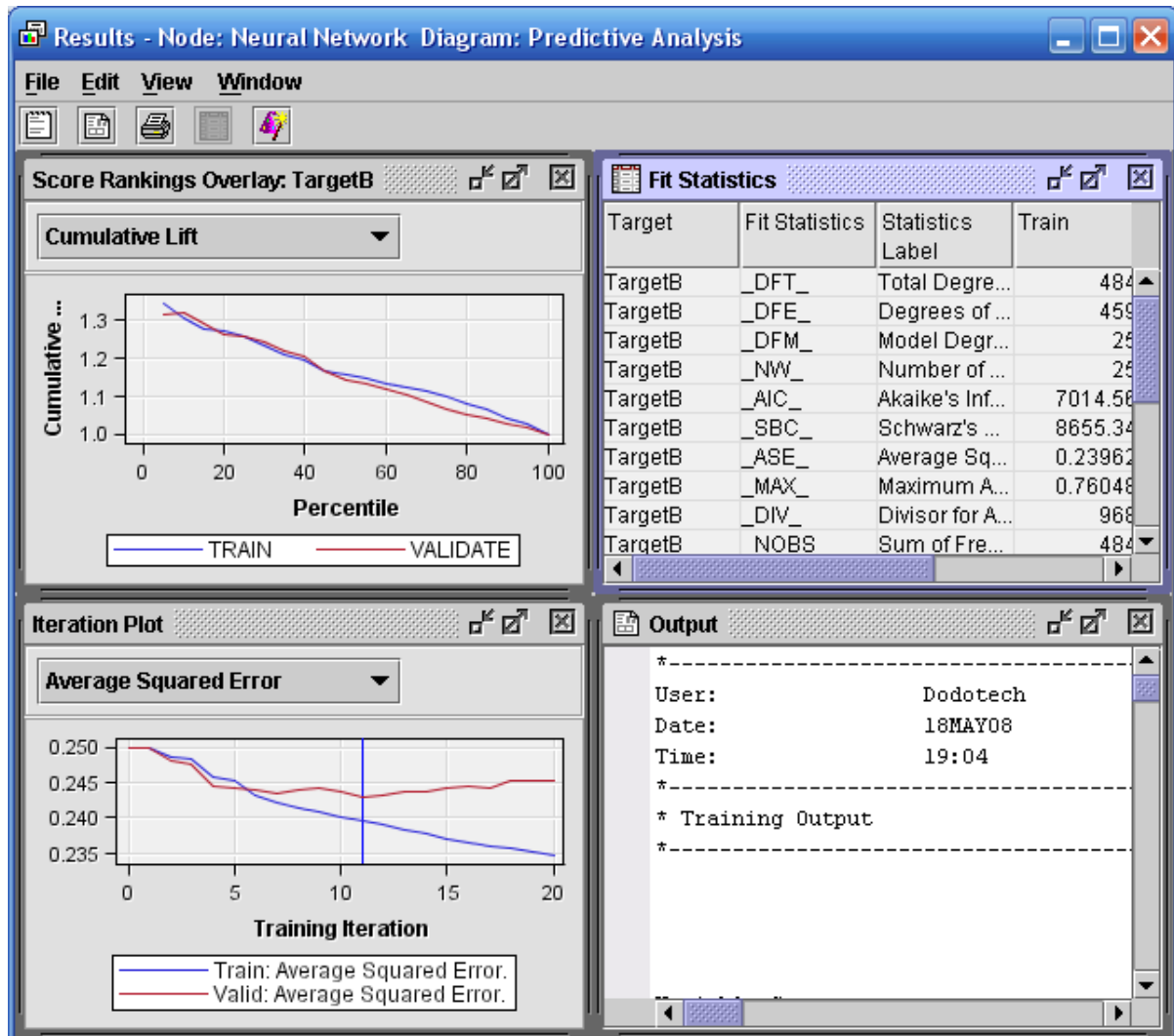
Property	Value
<b>General</b>	
Node ID	Neural
Imported Data	...
Exported Data	...
Notes	...
<b>Train</b>	
Variables	...
Continue Training	No
Network	...
Optimization	...
Initialization Seed	12345
Model Selection Criterion	Profit/Loss
Suppress Output	No
<b>Score</b>	
Hidden Units	No
Residuals	Yes
Standardization	No

5. Select **Enable** ⇨ **No** under the Preliminary Training options.



## 6. Run the Neural Network node and view the results.

The Results - Node: Neural Network Diagram window opens.



## 7. Maximize the Fit Statistics window.

Fit Statistics					
Target	Fit Statistics	Statistics Label	Train	Validation	Test
TargetB	_DFT_	Total Degre...	4843	.	.
TargetB	_DFE_	Degrees of ...	4590	.	.
TargetB	_DFM_	Model Degr...	253	.	.
TargetB	_NW_	Number of ...	253	.	.
TargetB	_AIC_	Akaike's Inf...	7014.564	.	.
TargetB	_SBC_	Schwarz's ...	8655.342	.	.
TargetB	_ASE_	Average Sq...	0.239625	0.242925	.
TargetB	_MAX_	Maximum A...	0.760482	0.774254	.
TargetB	_DIV_	Divisor for A...	9686	9686	.
TargetB	_NOBS_	Sum of Fre...	4843	4843	.
TargetB	_RASE_	Root Avera...	0.489515	0.492874	.
TargetB	_SSE_	Sum of Squ...	2321.008	2352.972	.
TargetB	_SUMWV_	Sum of Cas...	9686	9686	.
TargetB	_FPE_	Final Predic...	0.266041	.	.
TargetB	_MSE_	Mean Squa...	0.252833	0.242925	.
TargetB	_RFPE_	Root Final ...	0.515792	.	.
TargetB	_RMSE_	Root Mean ...	0.502825	0.492874	.
TargetB	_AVERR_	Average Err...	0.671956	0.678893	.
TargetB	_ERR_	Error Functi...	6508.564	6575.759	.
TargetB	_MISC_	Misclassific...	0.418542	0.430105	.
TargetB	_WRONG_	Number of ...	2027	2083	.

The average squared error and misclassification are similar to the values observed from regression models in the previous chapter. Notice that the model contains 253 weights. This is a large model.

8. Go to line 54 of the Output window. There you can find a table of the initial values for the neural network weights.

The NEURAL Procedure			
Optimization Start			
Parameter Estimates			
N	Parameter	Estimate	Gradient Objective Function
1	DemMedHomeValue_H11	-0.004746	0
2	DemPctVeterans_H11	-0.011042	0
3	GiftTimeFirst_H11	-0.026889	0
4	GiftTimeLast_H11	-0.024545	0
5	IMP_DemAge_H11	0.008120	0
6	IMP_LOG_GiftAvgCard36_H11	0.055146	0
7	IMP_REP_DemMedIncome_H11	-0.167987	0
8	LOG_GiftAvg36_H11	0.087440	0
9	LOG_GiftAvgAll_H11	0.063190	0
.	.	.	.
250	H11_TargetB1	0	-0.004814
251	H12_TargetB1	0	-0.000030480
252	H13_TargetB1	0	0.001641
253	BIAS_TargetB1	-0.000413	1.178583E-15

Despite the huge number of weights, the model shows no signs of overfitting.

9. Go to line 395. You can find a summary of the model optimization (maximizing the likelihood estimates of the model weights).

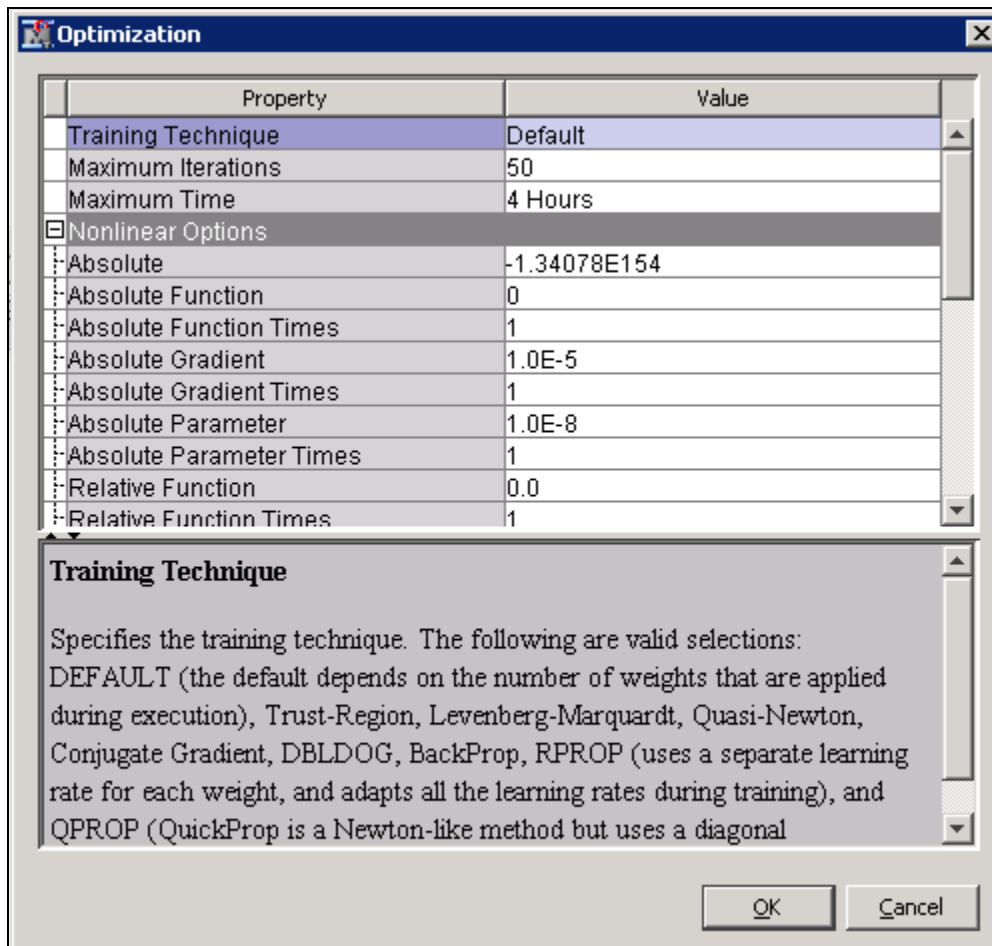
Optimization Results			
Iterations	50	Function Calls	136
Gradient Calls	62	Active Constraints	0
Objective Function	0.6368394579	Max Abs Gradient Element	0.0076065979
Slope of Search Direction	-0.000752598		
QUANEW needs more than 50 iterations or 2147483647 function calls.			
WARNING: QUANEW Optimization cannot be completed.			

Notice the warning message. It can be interpreted to mean that the model-fitting process did not converge.

10. Close the Results - Neural Network window.



11. Reopen the Optimization window and examine the Optimization options in the Properties panel for the Neural Network node.



The maximum number of iterations, by default, is 50. Apparently, this is not enough for the network training process to converge.

12. Type **100** for the Maximum Iterations property.
13. Run the Neural Network node and examine the results.
14. Maximize the Output window. You are again warned that the optimization process still failed to converge (even after 100 iterations).

QUANEW needs more than 100 iterations or 2147483647 function calls

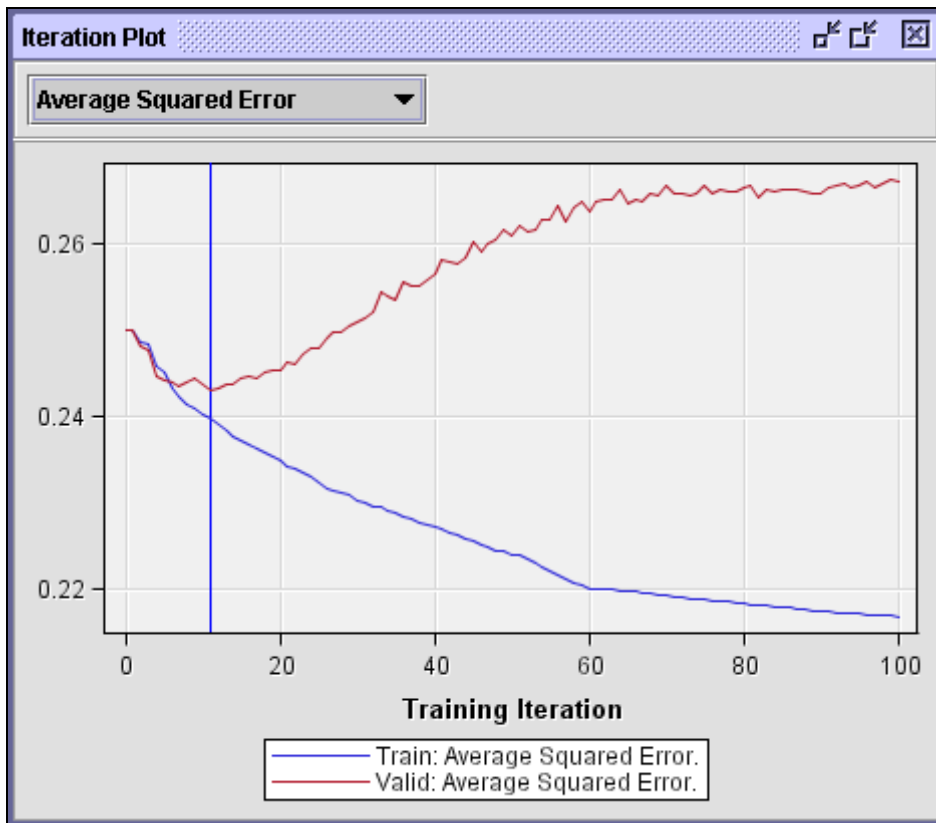
WARNING: QUANEW Optimization cannot be completed.

15. Maximize the Fit Statistics window.

Fit Statistics					
Target	Fit Statistics	Statistics Label	Train	Validation	Test
TargetB	_DFT_	Total Degre...	4843	.	.
TargetB	_DFE_	Degrees of ...	4590	.	.
TargetB	_DFM_	Model Degr...	253	.	.
TargetB	_NW_	Number of ...	253	.	.
TargetB	_AIC_	Akaike's Inf...	7014.564	.	.
TargetB	_SBC_	Schwarz's ...	8655.342	.	.
TargetB	_ASE_	Average Sq...	0.239625	0.242925	.
TargetB	_MAX_	Maximum A...	0.760482	0.774254	.
TargetB	_DIV_	Divisor for A...	9686	9686	.
TargetB	_NOBS_	Sum of Fre...	4843	4843	.
TargetB	_RASE_	Root Avera...	0.489515	0.492874	.
TargetB	_SSE_	Sum of Squ...	2321.008	2352.972	.
TargetB	_SUMW_	Sum of Cas...	9686	9686	.
TargetB	_FPE_	Final Predic...	0.266041	.	.
TargetB	_MSE_	Mean Squa...	0.252833	0.242925	.
TargetB	_RFPE_	Root Final ...	0.515792	.	.
TargetB	_RMSE_	Root Mean ...	0.502825	0.492874	.
TargetB	_AVERR_	Average Err...	0.671956	0.678893	.
TargetB	_ERR_	Error Functi...	6508.564	6575.759	.
TargetB	_MISC_	Misclassific...	0.418542	0.430105	.
TargetB	_WRONG_	Number of ...	2027	2083	.

Curiously, increasing the maximum number of iterations changes none of the fit statistics. How can this be? The answer is found in the Iteration Plot window.

16. Examine the Iteration Plot window.



The iteration plot shows the average squared error versus optimization iteration. A massive divergence in training and validation average squared error occurs near iteration 14, indicated by the vertical blue line.

The rapid divergence of the training and validation fit statistics is cause for concern. This primarily results from a huge number of weights in the fitted neural network model. The huge number of weights comes from the use of all inputs in the model. Reducing the number of modeling inputs reduces the number of modeling weights and possibly improves model performance.

17. Close the Results window.

## 5.2 Input Selection

### Model Essentials – Neural Networks

Predict new cases.	Prediction formula
<b>Select useful inputs.</b>	<b>None</b>
Optimize complexity.	Best model from sequence

19

The Neural Network tool in SAS Enterprise Miner lacks a built-in method for selecting useful inputs. While sequential selection procedures such as stepwise are known for neural networks, the computational costs of their implementation tax even fast computers. Therefore, these procedures are not part of SAS Enterprise Miner. You can solve this problem by using an external process to select useful inputs. In this demonstration, you use the variables selected by the standard regression model. (In Chapter 8, several other approaches are shown.)



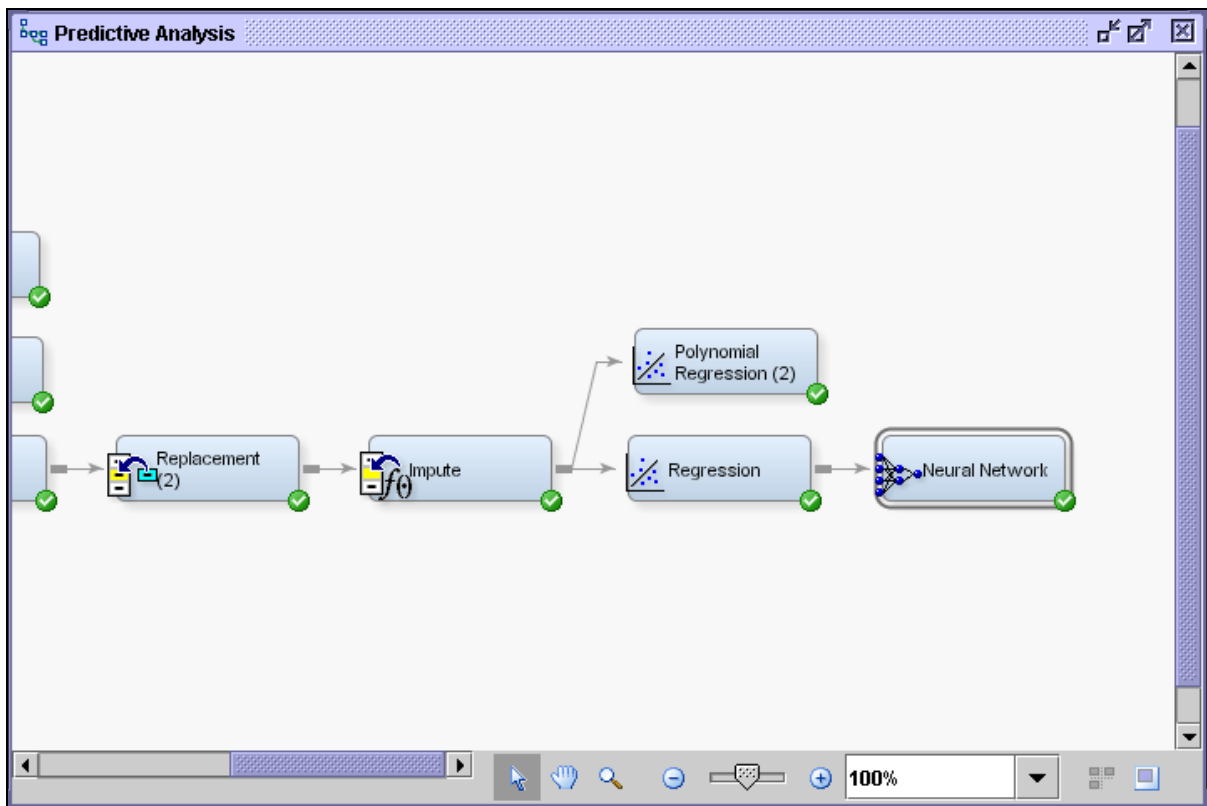
## Selecting Neural Network Inputs

This demonstration shows how to use a logistic regression to select inputs for a neural network.



Additional dimension reduction techniques are discussed in Chapter 8.

1. Delete the connection between the **Impute** node and the **Neural Network** node.
2. Connect the **Regression** node to the **Neural Network** node.



3. Right-click the **Neural Network** node and select **Update** from the shortcut menu.

- Open the Variables dialog box for the Neural Network node.

Name	Use	Report	Role	Level	Type	Order	Label	Format
DemCluster	Default	No	Rejected	Nominal	Character		Demographic	
DemGender	Default	No	Rejected	Nominal	Character		Gender	
DemHomeOw	Default	No	Rejected	Binary	Character		Home Owner	
DemMedHom	Default	No	Input	Interval	Numeric		Median Home	DOLLAR11
DemMedIncon	Default	No	Rejected	Interval	Numeric		Median Incom	DOLLAR11
DemPctVetera	Default	No	Rejected	Interval	Numeric		Percent Vetera	
GiftTimeFirst	Default	No	Rejected	Interval	Numeric		Times Since F	
GiftTimeLast	Default	No	Input	Interval	Numeric		Time Since La	
IMP_DemAge	Default	No	Rejected	Interval	Numeric		Imputed: Age	
IMP_LOG_Gift	Default	No	Rejected	Interval	Numeric		Imputed: Tran	
IMP_REP_Der	Default	No	Rejected	Interval	Numeric		Imputed: Repl	
LOG_GiftAvg3	Default	No	Rejected	Interval	Numeric		Transformed:	
LOG_GiftAvgAl	Default	No	Input	Interval	Numeric		Transformed:	
LOG_GiftAvgLa	Default	No	Rejected	Interval	Numeric		Transformed:	
LOG_GiftCnt3	Default	No	Input	Interval	Numeric		Transformed:	
LOG_GiftCntAl	Default	No	Rejected	Interval	Numeric		Transformed:	
LOG_GiftCntC	Default	No	Rejected	Interval	Numeric		Transformed:	
LOG_GiftCntC	Default	No	Rejected	Interval	Numeric		Transformed:	
M_DemAge	Default	No	Rejected	Binary	Numeric		Imputation Ind	
M_LOG_GiftAv	Default	No	Rejected	Binary	Numeric		Imputation Ind	
M_REP_Dem	Default	No	Rejected	Binary	Numeric		Imputation Ind	
PromCnt12	Default	No	Rejected	Interval	Numeric		Promotion Co	
PromCnt36	Default	No	Rejected	Interval	Numeric		Promotion Co	
PromCntAll	Default	No	Rejected	Interval	Numeric		Promotion Co	
PromCntCard	Default	No	Rejected	Interval	Numeric		Promotion Co	
PromCntCard	Default	No	Rejected	Interval	Numeric		Promotion Co	

Only the inputs selected by the Regression node's stepwise procedure are not rejected.

- Close the Variables dialog box.

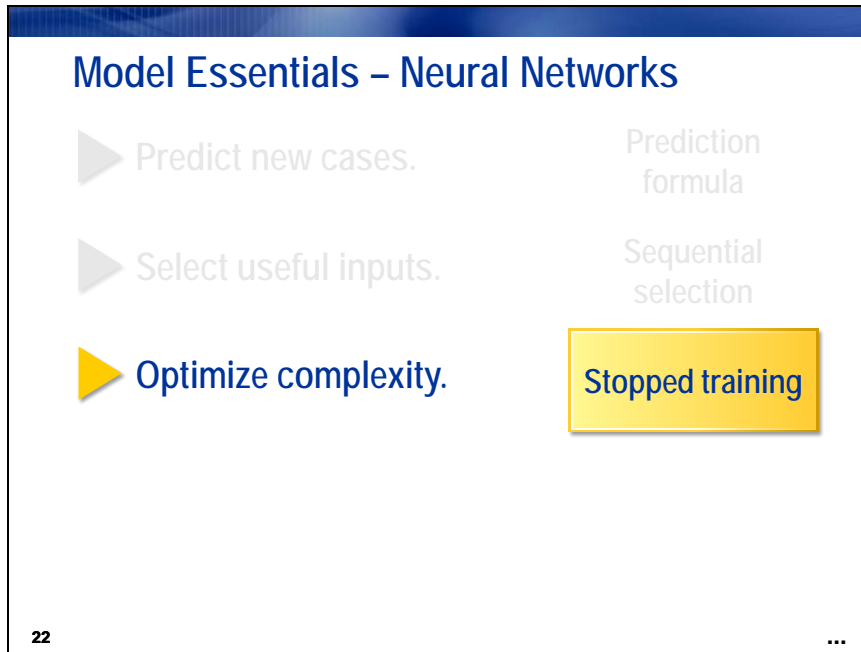
6. Run the Neural Network node and view the results.

The Fit Statistics window shows an improvement in model fit using only 19 weights.

Fit Statistics					
Target	Fit Statistics	Statistics Label	Train	Validation	Test
TargetB	_DFT_	Total Degre...	4843	.	.
TargetB	_DFE_	Degrees of ...	4824	.	.
TargetB	_DFM_	Model Degr...	19	.	.
TargetB	_NW_	Number of ...	19	.	.
TargetB	_AIC_	Akaike's Inf...	6573.409	.	.
TargetB	_SBC_	Schwarz's ...	6696.629	.	.
TargetB	_ASE_	Average Sq...	0.240983	0.240441	.
TargetB	_MAX_	Maximum A...	0.777	0.78162	.
TargetB	_DIV_	Divisor for A...	9686	9686	.
TargetB	_NOBS_	Sum of Fre...	4843	4843	.
TargetB	_RASE_	Root Avera...	0.4909	0.490348	.
TargetB	_SSE_	Sum of Squ...	2334.159	2328.913	.
TargetB	_SUMW_	Sum of Cas...	9686	9686	.
TargetB	_FPE_	Final Predic...	0.242881	.	.
TargetB	_MSE_	Mean Squa...	0.241932	0.240441	.
TargetB	_RFPE_	Root Final ...	0.49283	.	.
TargetB	_RMSE_	Root Mean ...	0.491866	0.490348	.
TargetB	_AVERR_	Average Err...	0.674727	0.673563	.
TargetB	_ERR_	Error Functi...	6535.409	6524.13	.
TargetB	_MISC_	Misclassific...	0.427421	0.421639	.
TargetB	_WRONG_	Number of ...	2070	2042	.

The validation and training average squared errors are nearly identical.

## 5.3 Stopped Training



The slide is titled "Model Essentials – Neural Networks" in blue text. It features three bullet points, each with a right-pointing triangle icon. The first two are greyed out: "Predict new cases." with "Prediction formula" to its right, and "Select useful inputs." with "Sequential selection" to its right. The third bullet point, "Optimize complexity.", is in blue and is preceded by a yellow triangle icon. To the right of this third bullet point is a yellow rectangular button with a black border and the text "Stopped training" in blue. In the bottom left corner, the number "22" is displayed, and in the bottom right corner, there are three dots "...".

Model Essentials – Neural Networks

- Predict new cases. Prediction formula
- Select useful inputs. Sequential selection
- Optimize complexity. **Stopped training**

22 ...

As was seen in the first demonstration, complexity optimization is an integral part of neural network modeling. Other modeling methods selected an optimal model from a sequence of possible models. In the demonstration, only one model was estimated (a neural network with three hidden units), so what was being compared?

SAS Enterprise Miner treats each iteration in the optimization process as a separate model. The iteration with the smallest value of the selected fit statistic is chosen as the final model. This method of model optimization is called *stopped training*.



## Fit Statistic versus Optimization Iteration

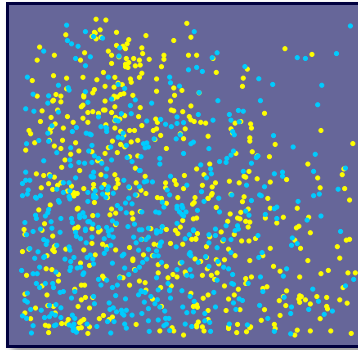
initial hidden unit weights

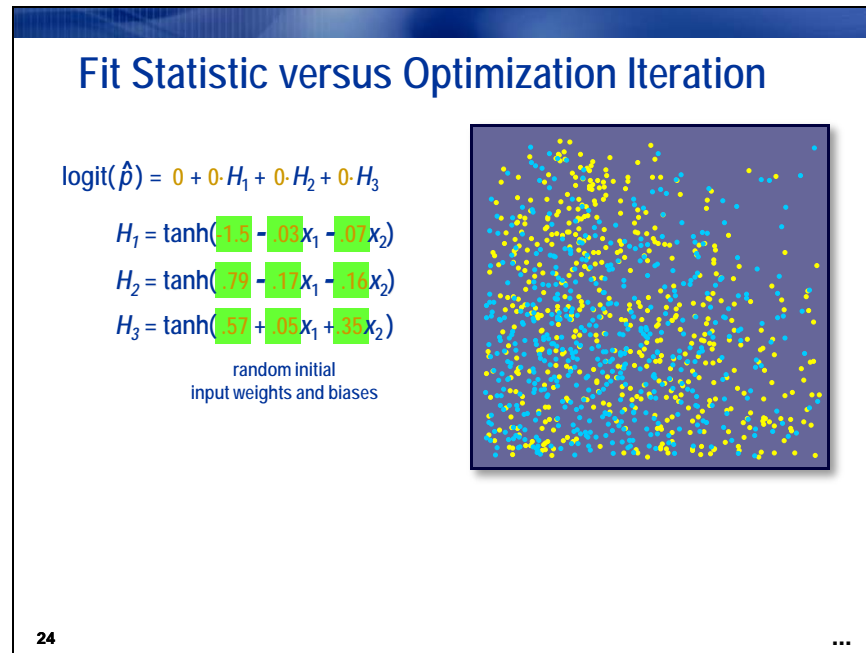
$$\text{logit}(\hat{p}) = 0 + 0H_1 + 0H_2 + 0H_3$$

$$H_1 = \tanh(-1.5 - .03x_1 - .07x_2)$$

$$H_2 = \tanh(.79 - .17x_1 - .16x_2)$$

$$H_3 = \tanh(.57 + .05x_1 + .35x_2)$$





To begin model optimization, model weights are given initial values. The weights multiplying the hidden units in the logit equation are set to zero, and the bias in the logit equation is set equal to the  $\text{logit}(\pi_1)$ , where  $\pi_1$  equals the primary outcome proportion. The remaining weights (corresponding to the hidden units) are given random initial values (near zero).

This “model” assigns each case a prediction estimate:  $\hat{p}_i = \pi_1$ . An initial fit statistic is calculated on training and validation data. For a binary target, this is proportional to the log likelihood function:

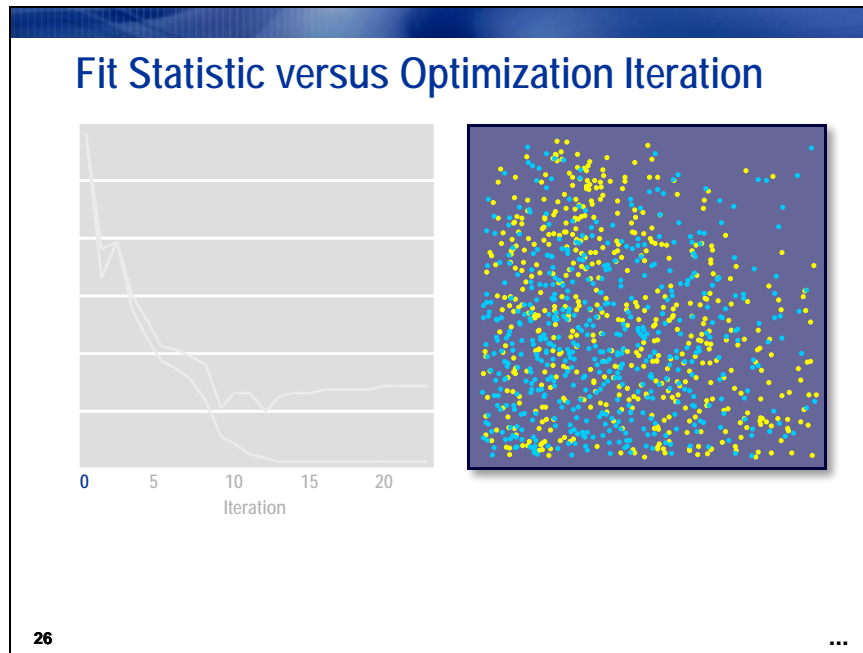
$$\sum_{\text{primary outcomes}} \log(\hat{p}_i(\hat{\mathbf{w}})) + \sum_{\text{secondary outcomes}} \log(1 - \hat{p}_i(\hat{\mathbf{w}}))$$

where

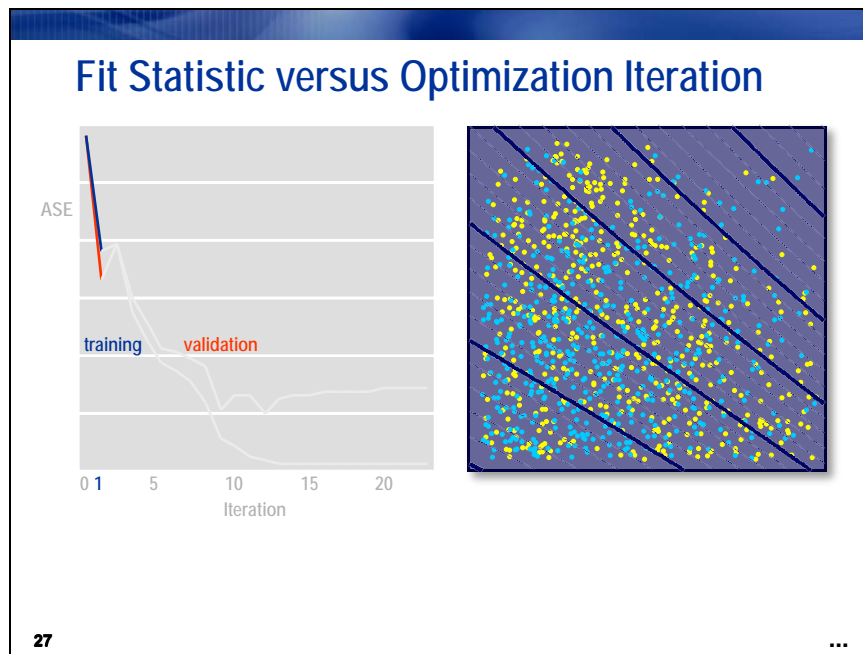
$\hat{p}_i$  is the predicted target value.

$\hat{\mathbf{w}}$  is the current estimate of the model parameters.

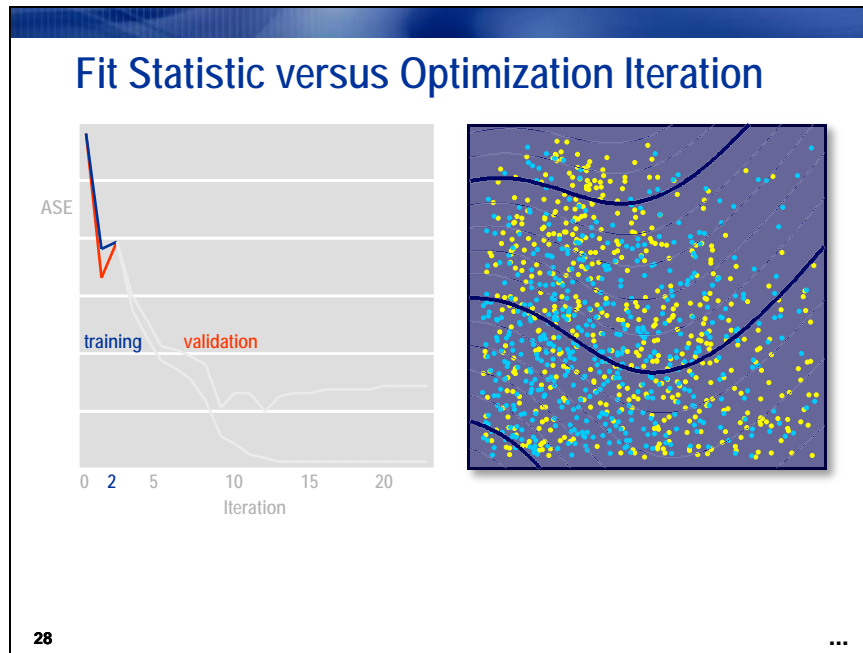
Training proceeds by updating the parameter estimates in a manner that decreases the value of the objective function.



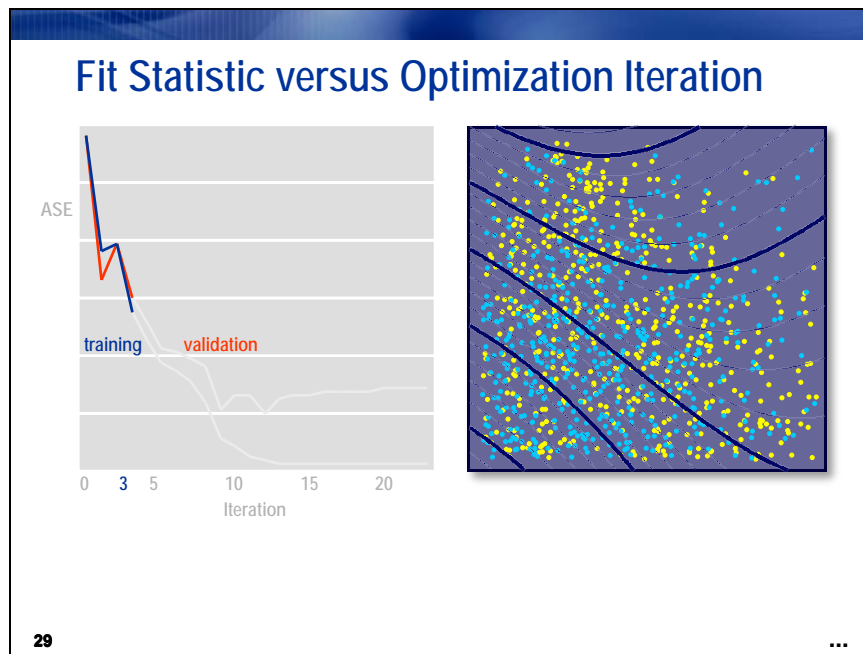
As stated above, in the initial step of the training procedure, the neural network model is set up to predict the overall average response rate for all cases.



One step substantially decreases the value average squared error (ASE). Amazingly, the model that corresponds to this one-iteration neural network looks very much like the standard regression model, as seen from the fitted isoclines.

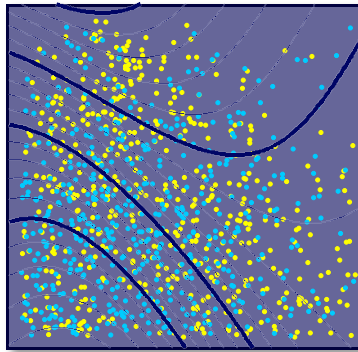
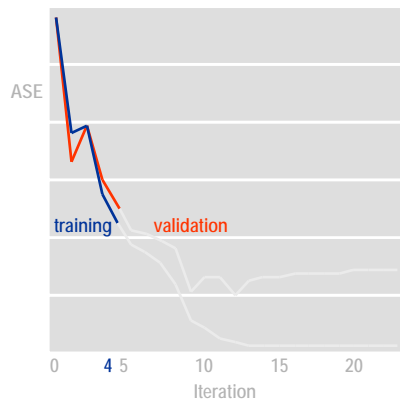


The second iteration step goes slightly astray. The model actually becomes slightly worse on the training and validation data.



Things are back on track in the third iteration step. The fitted model is already exhibiting nonlinear and nonadditive predictions. Half of the improvement in ASE is realized by the third step.

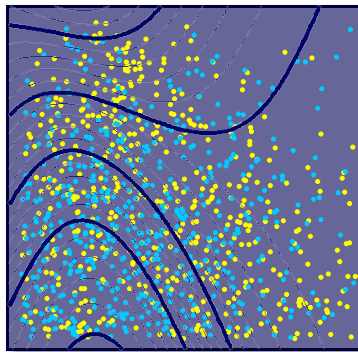
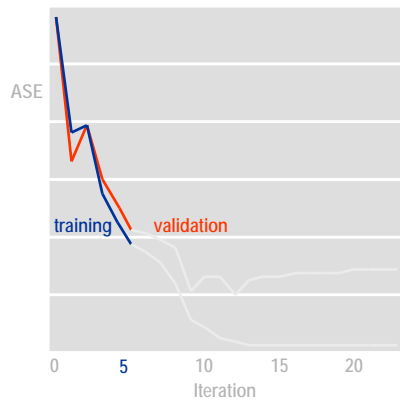
## Fit Statistic versus Optimization Iteration



30

...

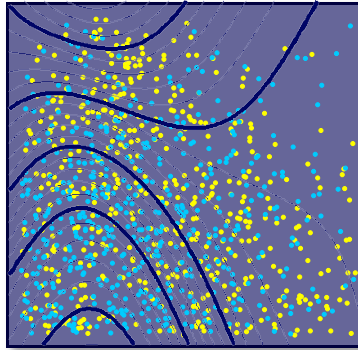
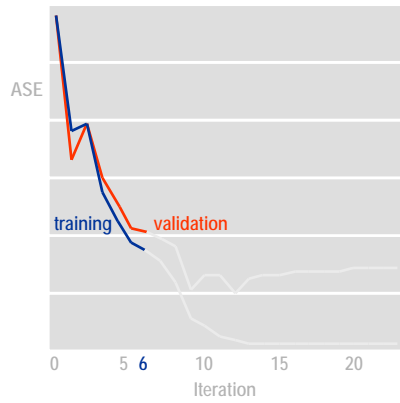
## Fit Statistic versus Optimization Iteration



31

...

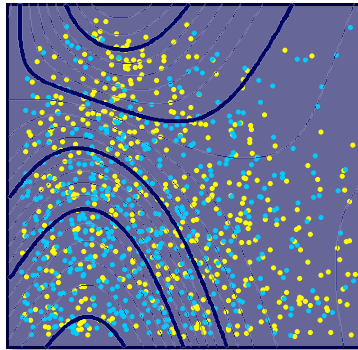
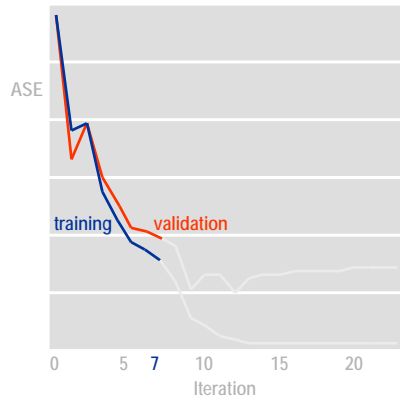
## Fit Statistic versus Optimization Iteration



32

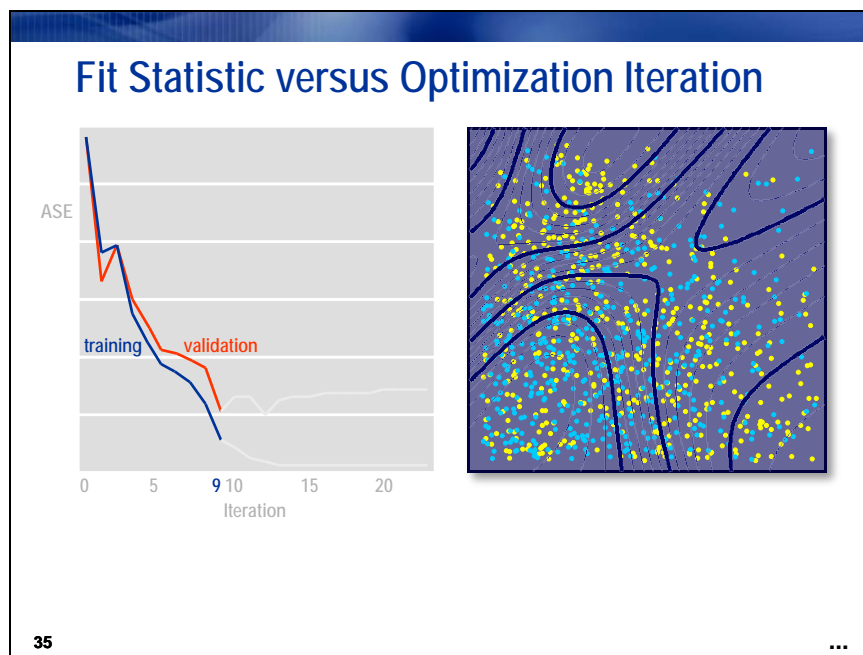
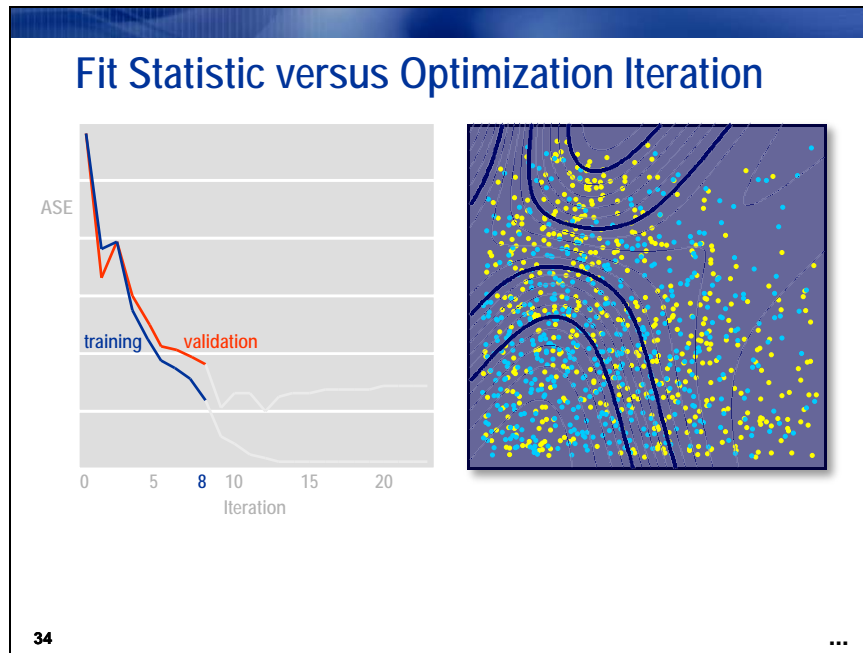
...

## Fit Statistic versus Optimization Iteration



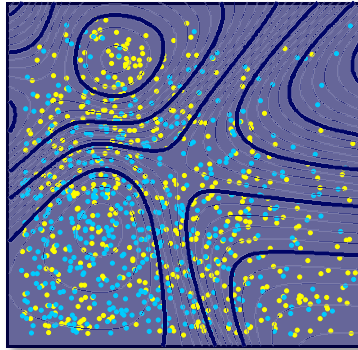
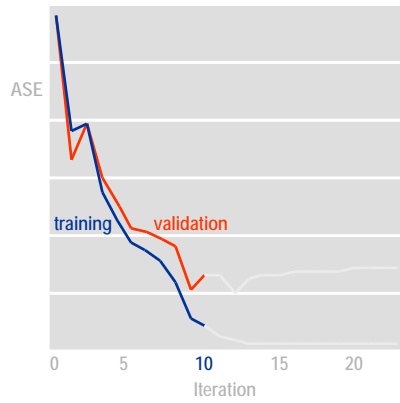
33

...



Most of the improvement in validation ASE occurred by the ninth step. (Training ASE continues to improve until convergence in Step 23.) The predictions are close to their final form.

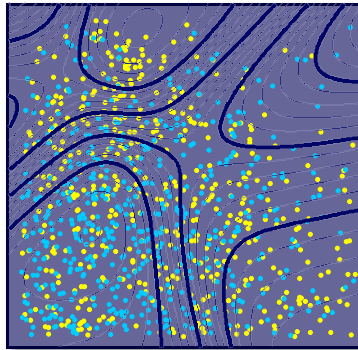
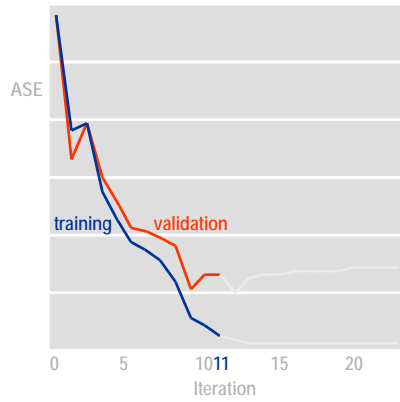
## Fit Statistic versus Optimization Iteration



36

...

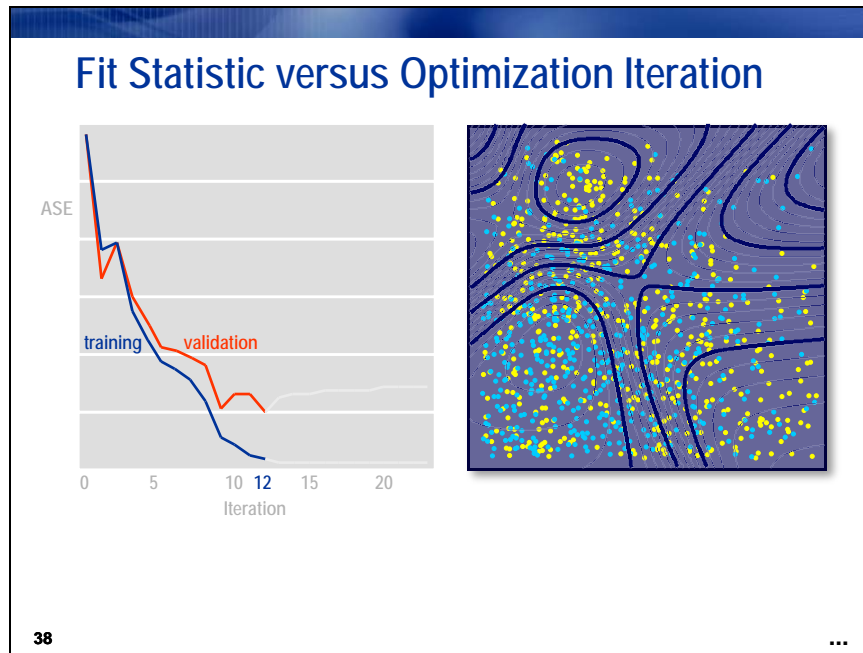
## Fit Statistic versus Optimization Iteration



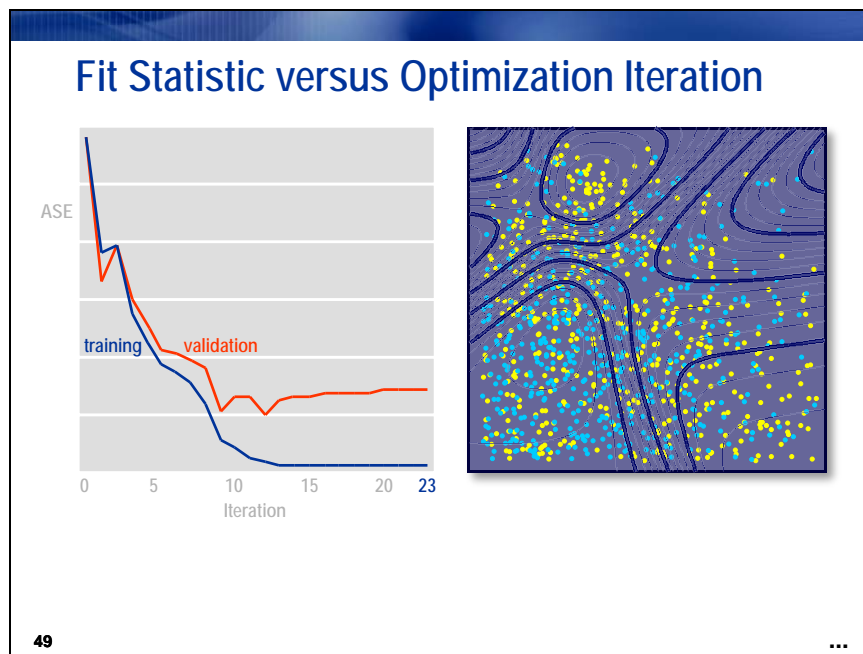
37

...

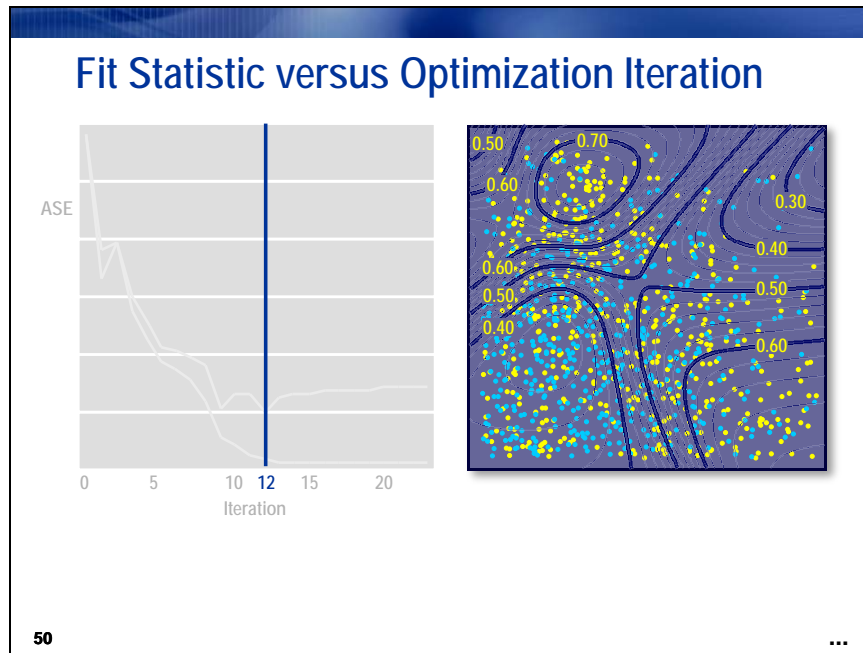




Step 12 brings the minimum value for validation ASE. While this model will ultimately be chosen as the final model, SAS Enterprise Miner continues to train until the likelihood objective function changes by a negligible amount on the training data.



In Step 23, training is declared complete due to lack of change in the objective function from Step 22. Notice that between Step 13 and Step 23, ASE actually increased for the validation data. This is a sign of overfitting.



The Neural Network tool selects the modeling weights from iteration 13 for the final model. In this iteration, the validation ASE is minimized. You can also configure the Neural Network tool to select the iteration with minimum misclassification or maximum profit for final weight estimates.



The name *stopped training* comes from the fact that the final model is selected as if training were stopped on the optimal iteration. Detecting when this optimal iteration occurs (while actually training) is somewhat problematic. To avoid stopping too early, the Neural Network tool continues to train until convergence on the training data or until reaching the maximum iteration count, whichever comes first.










## Increasing Network Flexibility

Stopped training helps to ensure that a neural network does not overfit (even when the number of network weights is large). Further improvement in neural network performance can be realized by increasing the number of hidden units from the default of three. There are two ways to explore alternative network sizes:

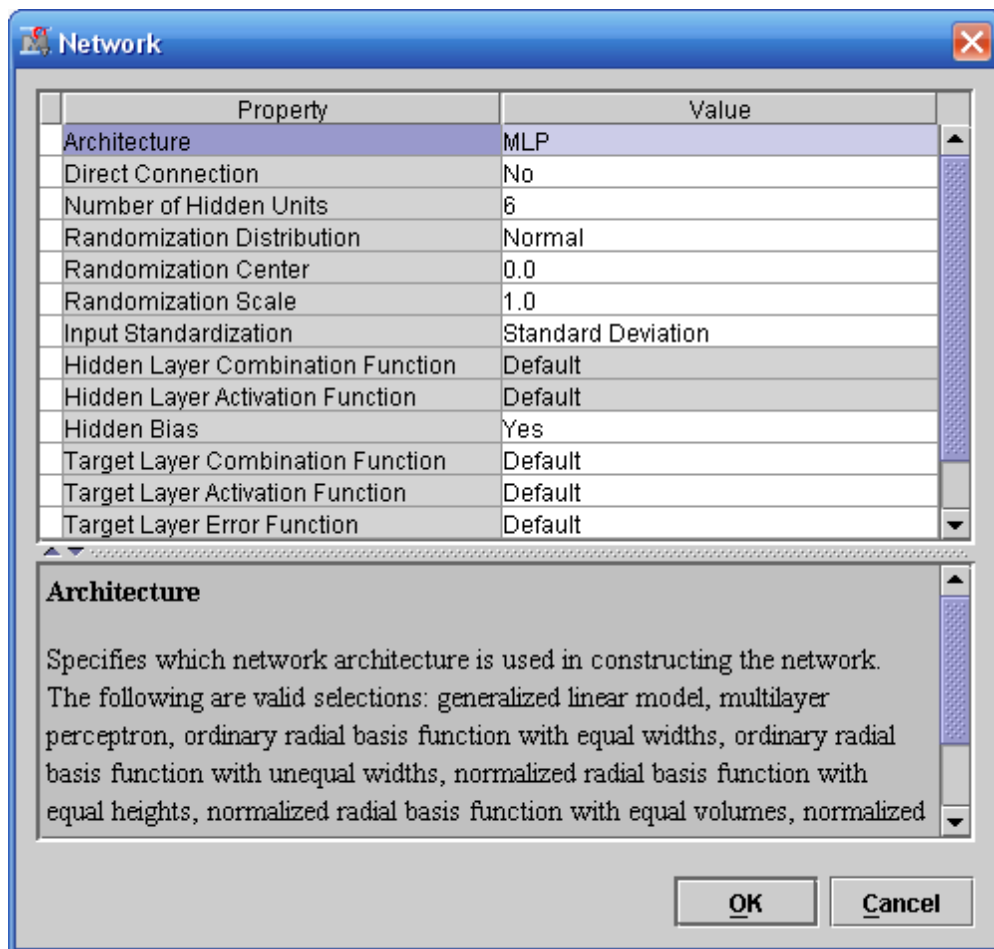
- manually, by changing the number of weights by hand
- automatically, by using the AutoNeural tool

Changing the number of hidden units manually involves trial-and-error guessing of the “best” number of hidden units. Several hidden unit counts were tried in advance. One of the better selections is demonstrated.

1. Select **Network** ⇨  from the Neural Network properties panel.

Property	Value
<b>General</b>	
Node ID	Neural
Imported Data	
Exported Data	
Notes	
<b>Train</b>	
Variables	
Continue Training	No
Network	
Optimization	
Initialization Seed	12345
Model Selection Crite	Profit/Loss
Suppress Output	No

The Network window opens.

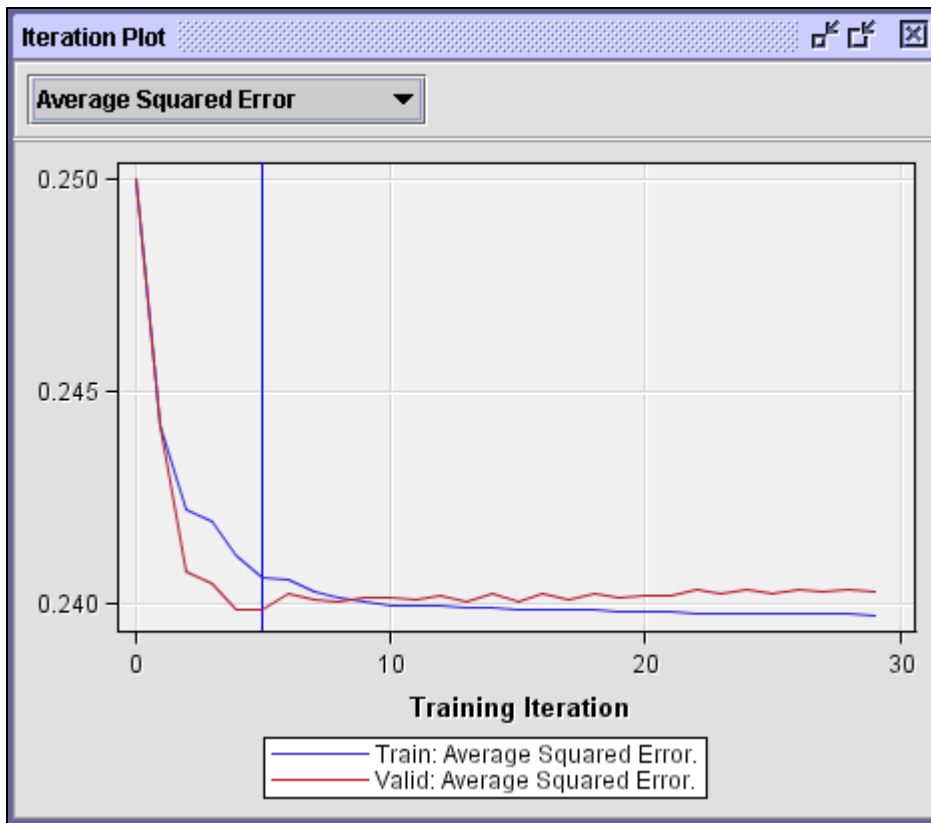


2. Type **6** as the Number of Hidden Units value.
3. Select **OK**.
4. Run the Neural Network node and view the results.

The Fit Statistics window shows good model performance on both the average squared error and misclassification scales.

Fit Statistics					
Target	Fit Statistics	Statistics Label	Train	Validation	Test
TargetB	_DFT_	Total Degre...	4843	.	.
TargetB	_DFE_	Degrees of ...	4806	.	.
TargetB	_DFM_	Model Degr...	37	.	.
TargetB	_NW_	Number of ...	37	.	.
TargetB	_AIC_	Akaike's Inf...	6601.6	.	.
TargetB	_SBC_	Schwarz's ...	6841.556	.	.
TargetB	_ASE_	Average Sq...	0.240623	0.23988	.
TargetB	_MAX_	Maximum A...	0.857328	0.869935	.
TargetB	_DIV_	Divisor for A...	9686	9686	.
TargetB	_NOBS_	Sum of Fre...	4843	4843	.
TargetB	_RASE_	Root Avera...	0.490533	0.489776	.
TargetB	_SSE_	Sum of Squ...	2330.673	2323.478	.
TargetB	_SUMW_	Sum of Cas...	9686	9686	.
TargetB	_FPE_	Final Predic...	0.244328	.	.
TargetB	_MSE_	Mean Squa...	0.242475	0.23988	.
TargetB	_RFPE_	Root Final ...	0.494295	.	.
TargetB	_RMSE_	Root Mean ...	0.492418	0.489776	.
TargetB	_AVERR_	Average Err...	0.673921	0.672391	.
TargetB	_ERR_	Error Functi...	6527.6	6512.78	.
TargetB	_MISC_	Misclassific...	0.427421	0.422878	.
TargetB	_WRONG_	Number of ...	2070	2048	.

The iteration plot shows optimal validation average squared error occurring on iteration 5.



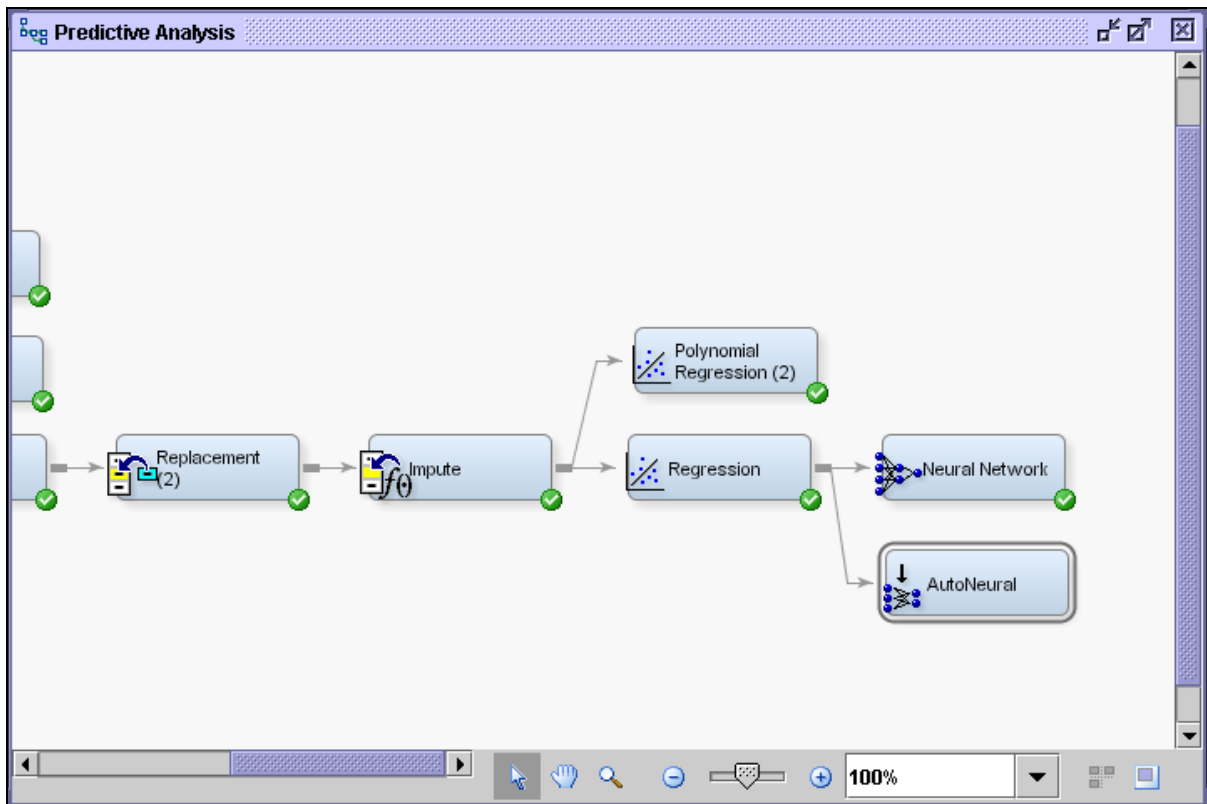
Unfortunately, there is little else of interest to describe this model. (In Chapter 8, a method is shown to give some insight into describing the cases with high primary outcome probability.)



## Using the AutoNeural Tool (Self-Study)

The AutoNeural tool offers an automatic way to explore alternative network architectures and hidden unit counts. This demonstration shows how to explore neural networks with increasing hidden unit counts.

1. Select the **Model** tab.
2. Drag the **AutoNeural** tool into the diagram workspace.
3. Connect the **Regression** node to the **AutoNeural** node as shown.



Six changes must be made to the AutoNeural node's default settings.

4. Select **Train Action** ⇒ **Search**. This configures the AutoNeural node to sequentially increase the network complexity.
5. Select **Number of Hidden Units** ⇒ **1**. With this option, each iteration adds one hidden unit.
6. Select **Tolerance** ⇒ **Low**. This prevents preliminary training from occurring.
7. Select **Direct** ⇒ **No**. This deactivates direct connections between the inputs and the target.
8. Select **Normal** ⇒ **No**. This deactivates the normal distribution activation function.

9. Select **Sine** ⇒ **No**. This deactivates the sine activation function.

Train	
Variables	
Model Options	
Architecture	Single Layer
Termination	Overfitting
Train Action	Search
Target Layer Error Function	Default
Maximum Iterations	8
Number of Hidden Units	1
Tolerance	Low
Total Time	One Hour
Increment and Search	
Adjust Iterations	Yes
Freeze Connections	No
Total Number of Hidden Units	30
Final Training	Yes
Final Iterations	5
Activation Functions	
Direct	No
Exponential	No
Identity	No
Logistic	No
Normal	No
Reciprocal	No
Sine	No
Softmax	No
Square	No
Tanh	Yes

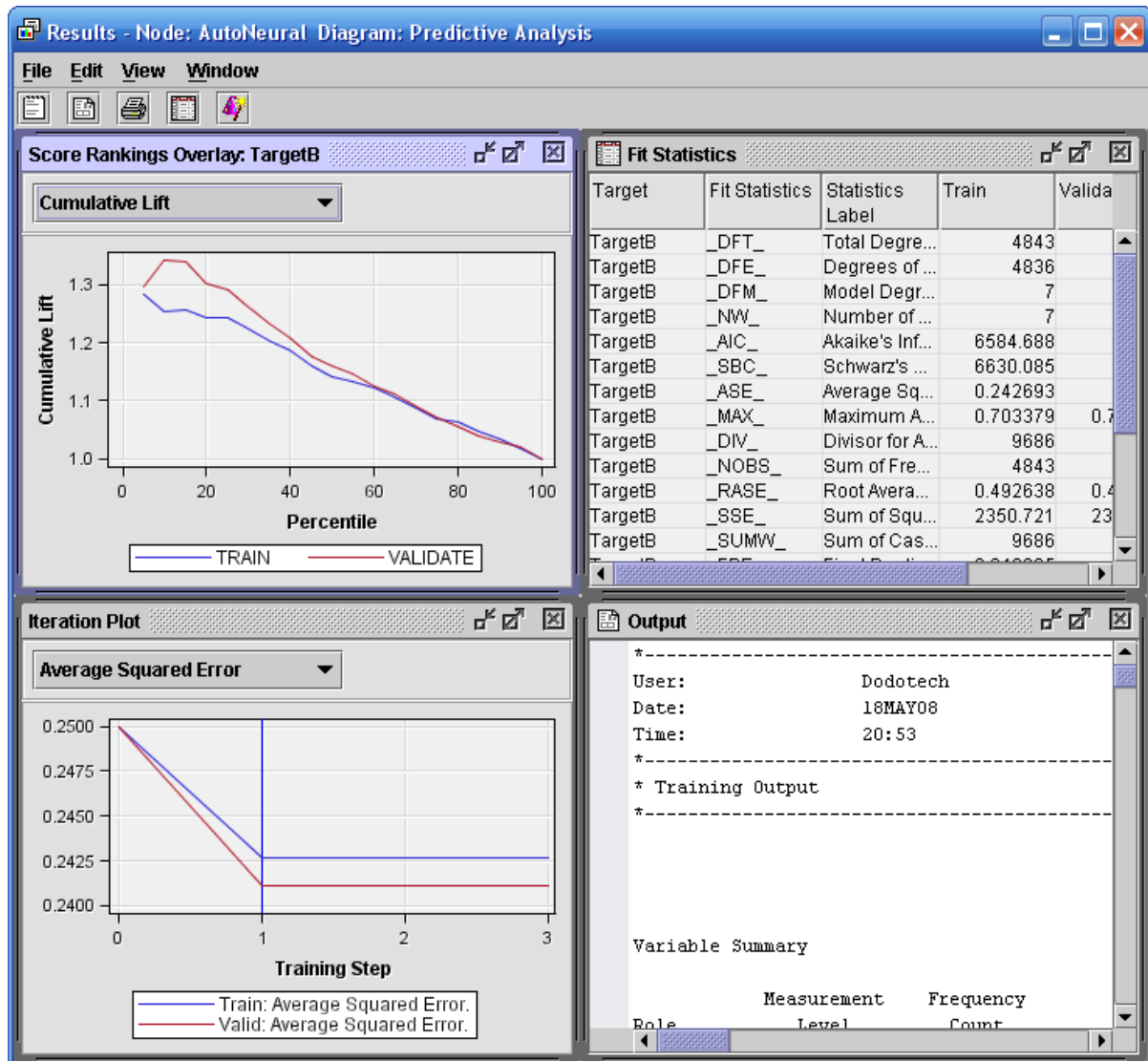
With these settings, each iteration adds one hidden unit to the neural network. Only the hyperbolic tangent activation function is considered.



After each iteration, the existing network weights are **not** reinitialized. With this restriction, the influence of additional hidden units decreases. Also, the neural network models that you obtain with the AutoNeural and Neural Network tools will be different, even if both networks have the same number of hidden units.



10. Run the AutoNeural node and view the results. The Results - Node: AutoNeural Diagram window opens.

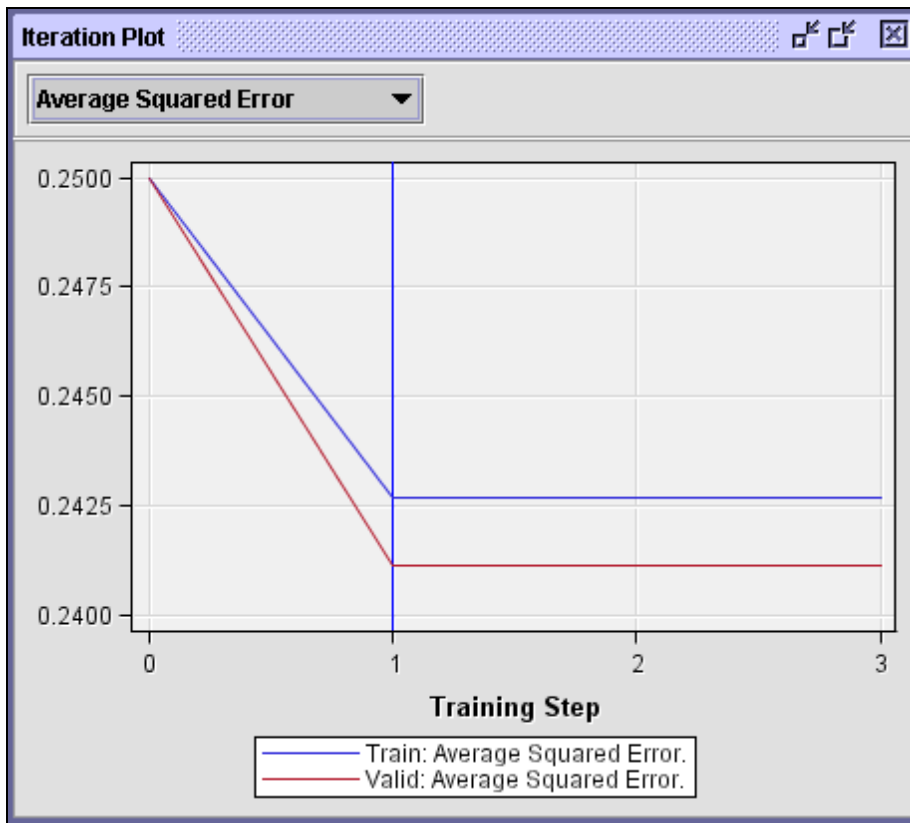


## 11. Maximize the Fit Statistics window.

Fit Statistics					
Target	Fit Statistics	Statistics Label	Train	Validation	Test
TargetB	_DFT_	Total Degre...	4843	.	.
TargetB	_DFE_	Degrees of ...	4836	.	.
TargetB	_DFM_	Model Degr...	7	.	.
TargetB	_NW_	Number of ...	7	.	.
TargetB	_AIC_	Akaike's Inf...	6584.688	.	.
TargetB	_SBC_	Schwarz's ...	6630.085	.	.
TargetB	_ASE_	Average Sq...	0.242693	0.2411	.
TargetB	_MAX_	Maximum A...	0.703379	0.705861	.
TargetB	_DIV_	Divisor for A...	9686	9686	.
TargetB	_NOBS_	Sum of Fre...	4843	4843	.
TargetB	_RASE_	Root Avera...	0.492638	0.491019	.
TargetB	_SSE_	Sum of Squ...	2350.721	2335.291	.
TargetB	_SUMW_	Sum of Cas...	9686	9686	.
TargetB	_FPE_	Final Predic...	0.243395	.	.
TargetB	_MSE_	Mean Squa...	0.243044	0.2411	.
TargetB	_RFPE_	Root Final ...	0.493351	.	.
TargetB	_RMSE_	Root Mean ...	0.492995	0.491019	.
TargetB	_AVERR_	Average Err...	0.67837	0.675163	.
TargetB	_ERR_	Error Functi...	6570.688	6539.625	.
TargetB	_MISC_	Misclassific...	0.426389	0.41751	.
TargetB	_WRONG_	Number of ...	2065	2022	.

The number of weights implies that the selected model has one hidden unit. The average squared error and misclassification rates are quite low.

12. Maximize the Iteration Plot window.



The AutoNeural and Neural Network node's iteration plots differ. The AutoNeural node's iteration plot shows the final fit statistic versus the number of hidden units in the neural network.

13. Maximize the Output window. The Output window describes the AutoNeural process.

14. Go to line 52.

Search # 1 SINGLE LAYER trial # 1 : TANH : Training					
_ITER_	_AIC_	_AVERR_	_MISC_	_VAVERR_	_VMISC_
0	6727.82	0.69315	0.49990	0.69315	0.50010
1	6725.17	0.69287	0.48462	0.69311	0.48193
2	6587.79	0.67869	0.42866	0.67713	0.42350
3	6584.69	0.67837	0.42639	0.67516	0.41751
4	6584.10	0.67831	0.42804	0.67638	0.43031
5	6575.69	0.67744	0.42660	0.67472	0.42061
6	6572.57	0.67712	0.42763	0.67455	0.42783
7	6571.21	0.67698	0.42866	0.67427	0.42205
8	6570.69	0.67692	0.42845	0.67420	0.42061
8	6570.69	0.67692	0.42845	0.67420	0.42061

These lines show various fit statistics versus training iteration using a single hidden unit network. Training stops at iteration 8 (based on an AutoNeural property setting). Validation misclassification is used to select the best iteration, in this case, Step 3. Weights from this iteration are selected for use in the next step.

## 15. View output lines 73-99.

Search # 2 SINGLE LAYER trial # 1 : TANH : Training					
_ITER_	_AIC_	_AVERR_	_MISC_	_VAVERR_	_VMISC_
0	6596.69	0.67837	0.42639	0.67516	0.41751
1	6587.08	0.67738	0.42866	0.67472	0.42639
2	6581.99	0.67685	0.42928	0.67405	0.42123
3	6580.65	0.67671	0.42887	0.67393	0.42391
4	6579.01	0.67654	0.42763	0.67392	0.42267
5	6578.27	0.67647	0.43011	0.67450	0.42804
6	6577.64	0.67640	0.42474	0.67426	0.42453
7	6577.57	0.67640	0.42680	0.67458	0.42825
8	6575.88	0.67622	0.42845	0.67411	0.42783
8	6575.88	0.67622	0.42845	0.67411	0.42783
Selected Iteration based on _VMISC_					
_ITER_	_AIC_	_AVERR_	_MISC_	_VAVERR_	_VMISC_
0	6596.69	0.67837	0.42639	0.67516	0.41751

A second hidden unit is added to the neural network model. All weights related to this new hidden unit are set to zero. All remaining weights are set to the values obtained in iteration 3 above. In this way, the two-hidden-unit neural network (Step 0) and the one-hidden-unit neural network (Step 3) have equal fit statistics.

Training of the two-hidden-unit network commences. The training process trains for eight iterations. Iteration 0 has the smallest validation misclassification and is selected to provide the weight values for the next AutoNeural step.

## 16. Go to line 106.

Final Training Training					
_ITER_	_AIC_	_AVERR_	_MISC_	_VAVERR_	_VMISC_
0	6584.69	0.67837	0.42639	0.67516	0.41751
1	6584.10	0.67831	0.42804	0.67638	0.43031
2	6573.21	0.67718	0.42783	0.67462	0.42350
3	6571.19	0.67698	0.42990	0.67437	0.42247
4	6570.98	0.67695	0.42887	0.67431	0.42308
5	6570.56	0.67691	0.43052	0.67418	0.42081
5	6570.56	0.67691	0.43052	0.67418	0.42081
Selected Iteration based on _VMISC_					
_ITER_	_AIC_	_AVERR_	_MISC_	_VAVERR_	_VMISC_
0	6584.69	0.67837	0.42639	0.67516	0.41751

The final model training commences. Again iteration zero offers the best validation misclassification.

The next block of output summarizes the training process. Fit statistics from the iteration with the smallest validation misclassification are shown for each step.

Final Training History									
_step_	_func_	_status_	_iter_	_AVERR_	_MISC_	_AIC_	_VAVERR_	_VMISC_	
SINGLE LAYER 1	TANH	initial	0	0.69315	0.49990	6727.82	0.69315	0.50010	
SINGLE LAYER 1	TANH	keep	3	0.67837	0.42639	6584.69	0.67516	0.41751	
SINGLE LAYER 2	TANH	reject	0	0.67837	0.42639	6596.69	0.67516	0.41751	
		Final	0	0.67837	0.42639	6584.69	0.67516	0.41751	
Final Model									
Stopping: Termination criteria was satisfied: overfitting based on _VMISC_									
_func_	_AVERR_	_VAVERR_	neurons						
TANH	0.67837	0.67516	1						
			=====						
			1						

The Final Model shows the hidden units added at each step and the corresponding value of the objective function (related to the likelihood).

## 5.4 Other Modeling Tools (Self-Study)

### Model Essentials – Rule Induction

▶ Predict new cases.	Prediction rules / prediction formula
▶ Select useful inputs.	Split search / none
▶ Optimize complexity.	Ripping / stopped training

55

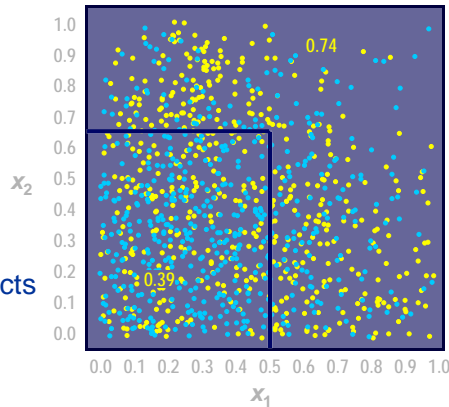
There are several additional modeling tools in SAS Enterprise Miner. This section describes the intended purpose (that is, when you should consider using them), how they perform the modeling essentials, and how they predict the simple example data.

The *Rule Induction tool* combines decision tree and neural network models to predict **nominal** targets. It is intended to be used when one of the nominal target levels is rare.

New cases are predicted using a combination of prediction rules (from decision trees) and a prediction formula (from a neural network, by default). Input selection and complexity optimization are described below.

## Rule Induction Predictions

- [Rips create prediction rules.]
- A binary model sequentially classifies and removes correctly classified cases.
- [A neural network predicts remaining cases.]



56

The Rule Induction algorithm has three steps:

- Using a decision tree, the first step attempts to locate “pure” concentrations of cases. These are regions of the input space containing only a single value of the target. The rules identifying these pure concentrations are recorded in the scoring code, and the cases in these regions are removed from the training data.

The simple example data does not contain any “pure” regions, so the step is skipped.

- The second step attempts to filter easy-to-classify cases. This is done with a sequence of binary target decision trees. The first tree in the sequence attempts to distinguish the most common target level from the others. Cases found in leaves correctly classifying the most common target level are removed from the training data. Using this revised training data, a second tree is built to distinguish the second most common target class from the others. Again, cases in any leaf correctly classifying the second most common target level are removed from the training data. This process continues through the remaining levels of the target.
- In the third step, a neural network is used to predict the remaining cases. All inputs selected for use in the Rule Induction node will be used in the neural network model. Model complexity is controlled by stopped training using classification as the fit statistic.

### Model Essentials – Dmine Regression

- ▶ Predict new cases. Prediction formula
- ▶ Select useful inputs. Forward selection
- ▶ Optimize complexity. Stop R-square

57

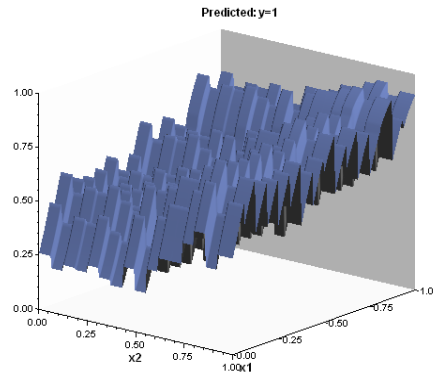
*Dmine regression* is designed to provide a regression model with more flexibility than a standard regression model. It should be noted that with increased flexibility comes an increased potential of overfitting.

A regression-like prediction formula is used to score new cases. Forward selection picks the inputs. Model complexity is controlled by a minimum R-squared statistic.



## Dmine Regression Predictions

- Interval inputs binned, categorical inputs grouped
- Forward selection picks from binned and original inputs



58

The main distinguishing feature of Dmine regression versus traditional regression is its grouping of categorical inputs and binning of continuous inputs.

- The levels of each categorical input are systematically grouped together using an algorithm that is reminiscent of a decision tree. Both the original and grouped inputs are made available for subsequent input selection.
- All interval inputs are broken into a maximum of 16 bins in order to accommodate nonlinear associations between the inputs and the target. The levels of the maximally binned interval inputs are grouped using the same algorithm for grouping categorical inputs. These binned-and-grouped inputs and the original interval inputs are made available for input selection.

A forward selection algorithm selects from the original, binned, and grouped inputs. Only inputs with an R square of 0.005 or above are eligible for inclusion in the forward selection process. Forward selection on eligible inputs stops when no input improves the R square of the model by the default value 0.0005.

## Model Essentials – DMNeural

▶ Predict new cases.

Stagewise  
prediction formula

▶ Select useful inputs.

Principal  
component

▶ Optimize complexity.

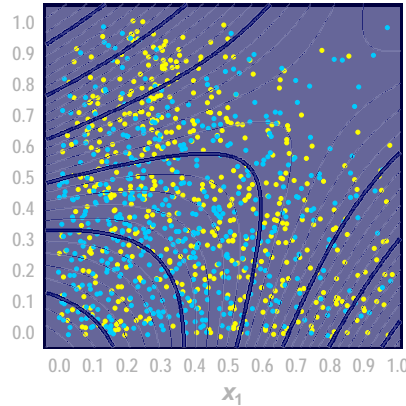
Max stage

59

The DMNeural tool is designed to provide a flexible target prediction using an algorithm with some similarities to a neural network. A multi-stage prediction formula scores new cases. The problem of selecting useful inputs is circumvented by a principal components method. Model complexity is controlled by choosing the number of stages in the multi-stage predictions formula.

## DMNeural Predictions

- Up to three PCs with highest target R square are selected.
- One of eight continuous transformations are selected and applied to selected PCs.
- The process is repeated three times with residuals from each stage.

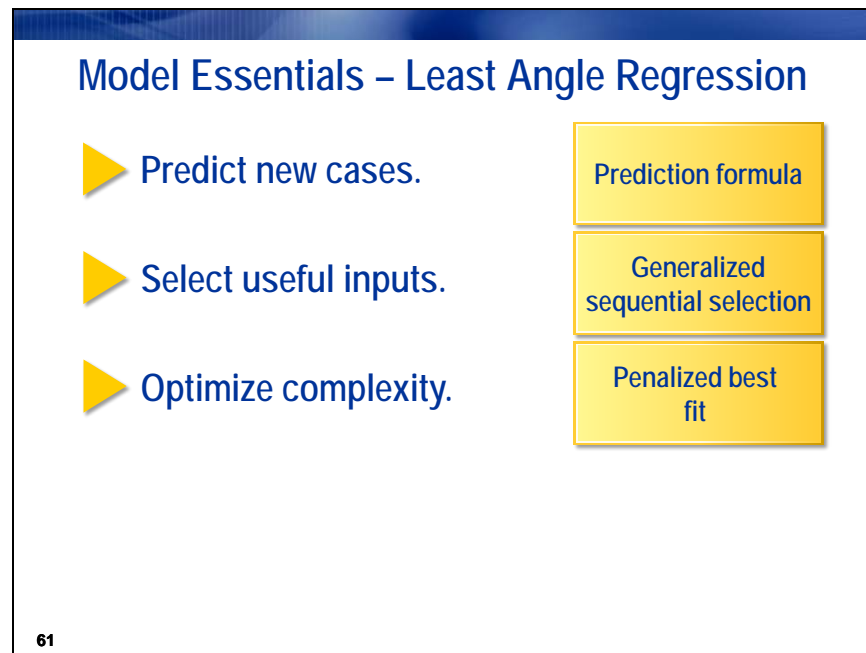


60

The algorithm starts by transforming the original inputs into principle components, which are orthogonal linear transformations of the original variables. The three principal components with the highest target correlation are selected for the next step.

Next, one of eight possible continuous transformations (square, hyperbolic tangent, arctangent, logistic, Gaussian, sine, cosine, exponential) is applied to the three principle component inputs. (For efficiency, the transformation is actually applied to a gridded version of the principle component, where each grid point is weighted by the number of cases near the point.) The transformation with the optimal fit statistic (squared error, by default) is selected. The target is predicted by a regression model using the selected principal components and transformation.

The process in the previous paragraph is repeated on the residual (the difference between the observed and predicted target value) up to the number of times specified in the maximum stage property (three, by default).



Forward selection, discussed in the previous chapter, provides good motivation for understanding how the Least Angle Regression (LARS) node works. In forward selection, the model-building process starts with an intercept. The best candidate input (the variable whose parameter estimate is most significantly different from zero) is added to create a one-variable model in the first step. The best candidate input variable from the remaining subset of input variables is added to create a two-variable model in the second step, and so on. Notice that an equivalent way to phrase the description of what happens in Step 2 is that the candidate input variable *most highly correlated with the residuals of the one variable model* is added to create a two-variable model in the second step.

The LARS algorithm generalizes forward selection in the following way:

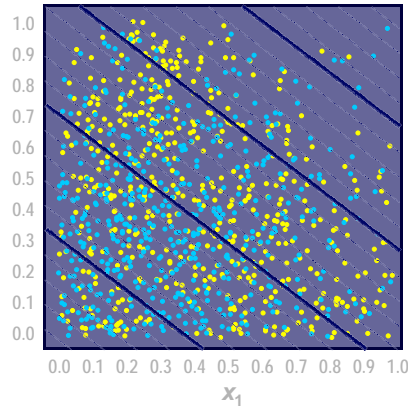
1. Weight estimates are initially set to a value of zero.
2. The slope estimate in the one variable model grows away from zero until some other candidate input has an equivalent correlation with the residuals of that model.
3. Growth of the slope estimate on the first variable stops, and growth on the slope estimate of the second variable begins.
4. This process continues until the least squares solutions for the weights are attained, or some stopping criterion is optimized.

This process can be constrained by putting a threshold on the aggregate magnitude of the parameter estimates. The LARS node provides an option to use the LASSO (Least Absolute Shrinkage and Selection Operator) method for variable subset selection.

The LARS node optimizes the complexity of the model using a penalized best fit criterion. The Schwarz's Bayesian Criterion (SBC) is the default.

## Least Angle Regression Predictions

- Inputs are selected using a generalization of forward selection.
- An input combination in the sequence with optimal, penalized validation assessment is selected by default.



62

The Least Angle Regression node can be useful in a situation where there are many candidate input variables to choose from and the data set is not large. As mentioned above, the best subset model is chosen using SBC by default. SBC and its cousins (AIC, AICC, and others) optimize complexity by penalizing the fit of candidate input variables. That is, for an input to enter the model, it must improve the overall fit (here, diminish the residual sum of squares) of the model enough to overcome a penalty term built into the fit criterion. Other criteria, based on predicted (simulated) residual sums of squares or a validation data set, are available.

### Model Essentials – MBR

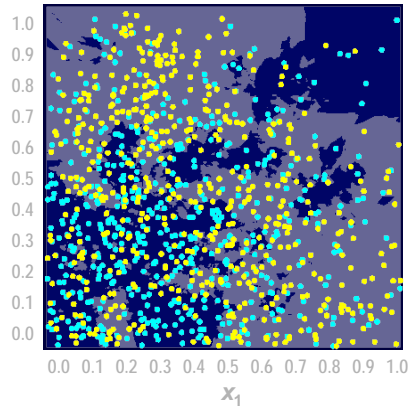
▶ Predict new cases.	Training data nearest neighbors
▶ Select useful inputs.	None
▶ Optimize complexity.	Number of neighbors

63

Memory Based Reasoning (MBR) is the implementation of  $k$ -nearest neighbor prediction in SAS Enterprise Miner. MBR predictions should be limited to decisions rather than rankings or estimates. Decisions are made based on the prevalence of each target level in the nearest  $k$ -cases (16 by default). A majority of primary outcome cases results in a primary decision, and a majority of secondary outcome cases results in a secondary decision. Nearest neighbor algorithms have no means to select inputs and can easily fall victim to the curse of dimensionality. Complexity of the model can be adjusted by changing the number of neighbors considered for prediction.

## MBR Prediction Estimates

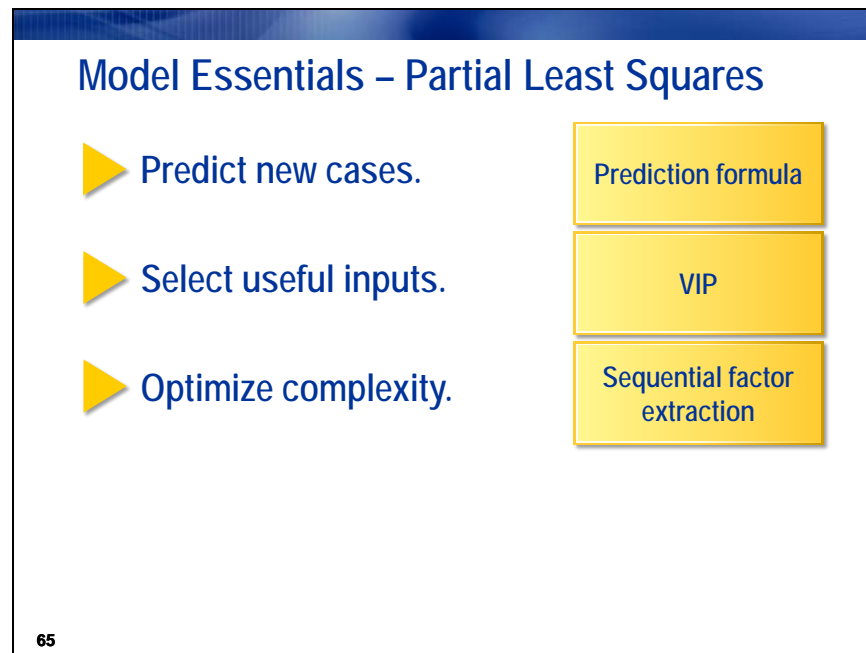
- Sixteen nearest training data cases predict the target for each point in the input space.
- Scoring requires training data and the PMBR procedure.



64

By default, the prediction decision made for each point in the input space is determined by the target values of the 16 nearest training data cases. Increasing the number of cases involved in the decision tends to smooth the prediction boundaries, which can be quite complex for small neighbor counts.

Unlike other prediction tools where scoring is performed by DATA step functions independent of the original training data, MBR scoring requires both the original training data set and a specialized SAS procedure, PMBR.



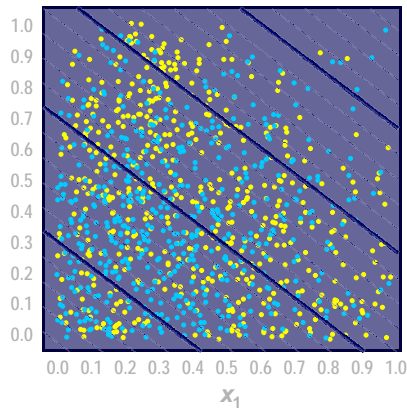
In partial least squares regression (PLS) modeling, model weights are chosen to simultaneously account for variability in both the target and the inputs. In general, model fit statistics like average squared error will be worse for a PLS model than for a standard regression model using the same inputs. A distinction of the PLS algorithm, however, is its ability to produce meaningful predictions based on limited data (for example, when there are more inputs than modeling cases). A variable selection method named *variable importance* in the projection arises naturally from the PLS framework.

In PLS, regression latent components, consisting of linear combinations of original inputs, are sequentially added. The final model complexity is chosen by optimizing a model fit statistic on validation data.



## Partial Least Squares Predictions

- Input combinations (factors) that optimally account for both predictor and response variation are successively selected.
- Factor count with a minimum validation PRESS statistic is selected.
- Inputs with small VIP are rejected for subsequent diagram nodes.



66

The PLS predictions look similar to a standard regression model. In the algorithm, input combinations, called *factors*, which are correlated with both input and target distributions, are selected. This choice is based on a projection given in the SAS/STAT PLS procedure documentation. Factors are sequentially added. The factor count with the lowest validation PRESS statistic (or equivalently, average squared error) is selected. In the PLS tool, the variable importance in the projection (VIP) for each input is calculated. Inputs with VIP less than 0.8 are rejected for subsequent nodes in the process flow.



## Exercises

### 1. Predictive Modeling Using Neural Networks

- a. In preparation for a neural network model, is imputation of missing values needed? \_\_\_\_\_  
Why or why not? \_\_\_\_\_
- b. In preparation for a neural network model, is data transformation generally needed? \_\_\_\_\_  
Why or why not? \_\_\_\_\_
- c. Add a Neural Network tool to the Organics diagram. Connect the **Impute** node to the **Neural Network** node.
- d. Set the model selection criterion to average squared error.
- e. Run the Neural Network node and examine the validation average squared error.  
How does it compare to other models? \_\_\_\_\_

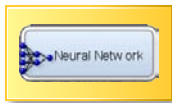
## 5.5 Chapter Summary

Neural networks are a powerful extension of standard regression models. They have the ability to model virtually any association between a set of inputs and a target. Being regression-like models, they share some of the same prediction complications faced by regression models (missing values, categorical inputs, extreme values). Their performance is reasonable even without input selection, but it can be marginally improved by selecting inputs with an external variable selection process. Most of their predictive success comes from stopped training, a technique used to prevent overfitting.

Network complexity is marginally influenced by the number of hidden units selected in the model. This number can be selected manually with the Neural Network tool, or it can be selected automatically using the AutoNeural tool.

SAS Enterprise Miner includes several other flexible modeling tools. These specialized tools include Rule Induction, Dmine Regression, DM Neural, and MBR.

### Neural Network Tool Review



Create a multi-layer perceptron on selected inputs. Control complexity with stopped training and hidden unit count.

## 5.6 Solutions

### Solutions to Exercises

#### 1. Predictive Modeling Using Neural Networks

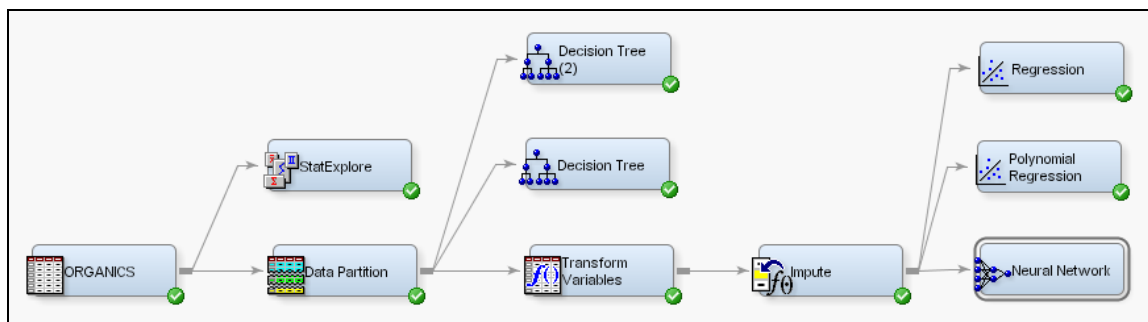
- a. In preparation for a neural network model, is imputation of missing values needed? **Yes.**

Why or why not? **Neural network models, as well as most models relying on a prediction formula, require a complete record for both modeling and scoring.**

- b. In preparation for a neural network model, is data transformation generally needed? **Not necessarily.**

Why or why not? **Neural network models create transformations of inputs for use in a regression-like model. However, having input distributions with low skewness and kurtosis tends to result in more stable models.**

- c. Add a Neural Network tool to the Organics diagram. Connect the **Impute** node to the **Neural Network** node.

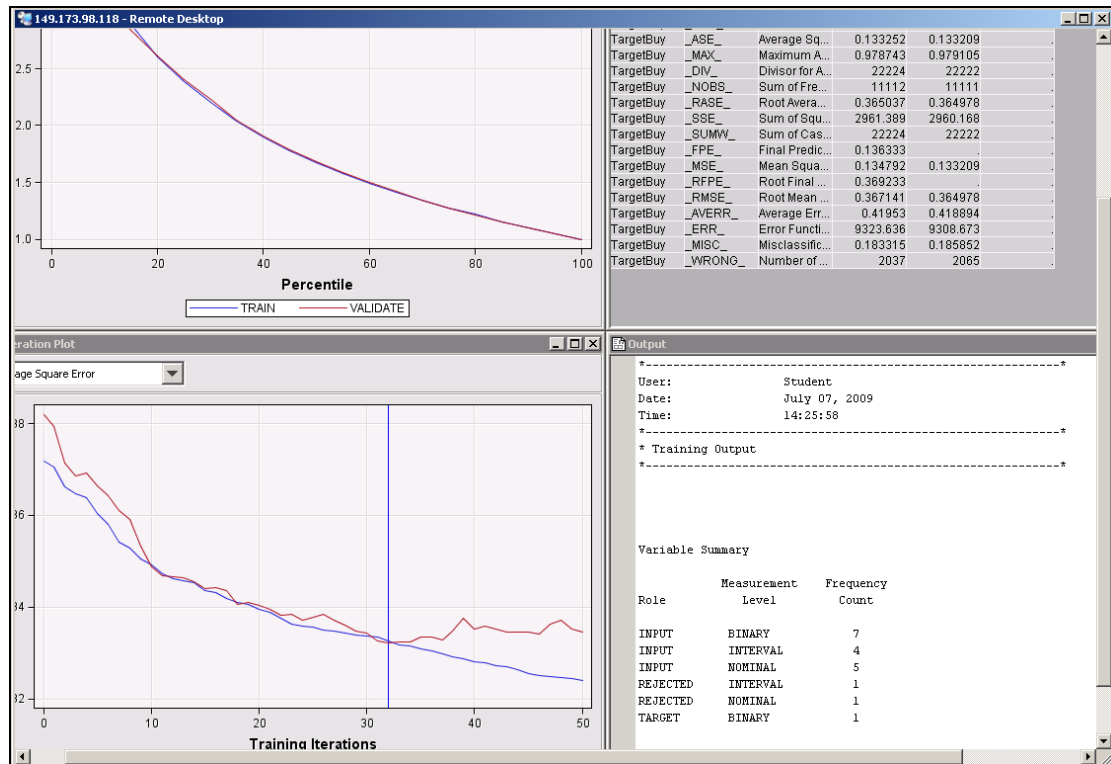


- d. Set the model selection criterion to average squared error.

Property	Value
<b>General</b>	
Node ID	Neural
Imported Data	...
Exported Data	...
Notes	...
<b>Train</b>	
Variables	...
Network	...
Model Selection Criterion	Average Error
Use Current Estimates	No
<input type="checkbox"/> Train Options	
Maximum Iterations	20
Maximum Time	4 Hours
Training Technique	Default
<input type="checkbox"/> Preliminary Training Options	
Preliminary Training	Yes
Maximum Iterations	10
Maximum Time	1 Hour
Number of Runs	5
<input type="checkbox"/> Convergence Criteria	
Uses Defaults	Yes
Options	...
<input type="checkbox"/> Print Options	
Suppress Output	No

e. Run the Neural Network node and examine the validation average squared error.

1) The Result window should appear as shown below.



2) Examine the Fit Statistics window.

Fit Statistics				
Target	Fit Statistics	Statistics Label	Train	Validation
TargetBuy	_DFT_	Total Degre...	11112	.
TargetBuy	_DFE_	Degrees of ...	10985	.
TargetBuy	_DFM_	Model Degr...	127	.
TargetBuy	_NW_	Number of ...	127	.
TargetBuy	_AIC_	Akaike's Inf...	9577.636	.
TargetBuy	_SBC_	Schwarz's ...	10506.74	.
TargetBuy	_ASE_	Average Sq...	0.133252	0.133209
TargetBuy	_MAX_	Maximum A...	0.978743	0.979105
TargetBuy	_DIV_	Divisor for A...	22224	22222
TargetBuy	_NOBS_	Sum of Fre...	11112	11111
TargetBuy	_RASE_	Root Avera...	0.365037	0.364978
TargetBuy	_SSE_	Sum of Squ...	2961.389	2960.168
TargetBuy	_SUMWV_	Sum of Cas...	22224	22222
TargetBuy	_FPE_	Final Predic...	0.136333	.
TargetBuy	_MSE_	Mean Squa...	0.134792	0.133209
TargetBuy	_RFPE_	Root Final ...	0.369233	.
TargetBuy	_RMSE_	Root Mean ...	0.367141	0.364978
TargetBuy	_AVERR_	Average Err...	0.41953	0.418894
TargetBuy	_ERR_	Error Functi...	9323.636	9308.673
TargetBuy	_MISC_	Misclassific...	0.183315	0.185852
TargetBuy	_WRONG_	Number of ...	2037	2065

How does it compare to other models? **The validation ASE for the neural network model is slightly smaller than the standard regression model, nearly the same as the polynomial regression, and slightly larger than the decision tree's ASE.**