


7. The AutoNeural Node (Self-Study)

7.1	Introduction.....	3
7.2	Architectures	12
	Demonstration: Comparing AutoNeural Architectures.....	18
7.3	Chapter Summary.....	23
7.4	Solutions	24
	Solutions to Student Activities (Polls/Quizzes).....	24

7.1 Introduction

THE POWER TO KNOW.

Objectives

- Detail the options available on the AutoNeural node's properties panel.

3

AutoNeural Node

General Properties

Property	Value
General	
Node ID	AutoNeural
Imported Data	...
Exported Data	...
Notes	...

Scoring Properties

Score	
Hidden Units	No
Residuals	Yes
Standardization	No

Training Properties

Train	
Variables	...
Model Options	
Architecture	Single Layer
Termination	Overfitting
Train Action	Search
Target Layer Error Function	Default
Maximum Iterations	8
Number of Hidden Units	2
Tolerance	Medium
Total Time	One Hour
Increment and Search Opti	
Adjust Iterations	Yes
Freeze Connections	No
Total Number of Hidden Uni	30
Final Training	Yes
Final Iterations	5
Activation Functions	
Direct	Yes
Exponential	No
Identity	No
Logistic	No
Normal	Yes
Reciprocal	No
Sine	Yes
Softmax	No
Square	No
Tanh	Yes

4

The AutoNeural node can be used to automatically construct a neural network. The code that it generates can also serve as the starting point for the development of custom neural network architectures.

Here are some of the key features of the AutoNeural node:

- Multiple hidden layer MLP networks can be constructed. Hidden layers are added one at a time.
- Preliminary training can be performed, if chosen.
- All specified combination and activation functions are tried.
- Incremental training. (Subsequent training is initialized with the current weight estimates.)
- Limited searches for the best network configuration.
- Complete control over the target layer error function is applied.
- Layers can be frozen during subsequent training.
- In each training iteration, one estimate vector and one fit vector are retained.

The AutoNeural properties panel is divided into sections. First is the **General** properties section, which contains the following information:

- | | |
|---------------|---|
| Node ID | The first AutoNeural node added to a diagram will have a Node ID value of AutoNeural. The second AutoNeural node added will have a Node ID of AutoNeural2, and so on. |
| Imported Data | The Imported Data - AutoNeural window contains a list of the ports that provide data sources to the AutoNeural node. |
| Exported Data | The Exported Data - AutoNeural window contains a list of the output data ports that the AutoNeural node creates data for when it runs. |
| Notes | Select the button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information. |

Next is the **Train** properties section of the AutoNeural node's properties panel. It has three subsections:

(a) Model Options

The Model Options subsection enables you to configure the general properties of your network. The available settings are described below.

Architecture	specifies the neural network architecture that you want to use to build the network. The default is the single layer architecture.
Termination	identifies the method that you want to use to end training. The default is to stop when overfitting is detected.
Train Action	specifies the training method that you want to use. The default is to use all selected activation functions until the stopping criterion is achieved.
Target Layer Error Function	specifies the error function in a user-defined network. For discrete target variables, the default error function is multiple Bernoulli. For interval targets, the default error function is a normal distribution.
Maximum Iterations	specifies the maximum number of iterations permitted during training. Permissible values are any integer between 1 and 50. The default is 8.
Number of Hidden Units	specifies the number of hidden units that you want to use. Permissible values are integers between 1 and 8. The default is 2.
Tolerance	configures the extent of the preliminary search. The default is Medium, which invokes preliminary training but does not apply the ABSCONV=.001 convergence criteria.
Total Time	specifies the amount of time allowed for training. The default is one hour.

(b) Increment and Search

When the Search Train option is specified, nodes are added according to the architecture. The best network is retained as the final model. The following options are available:

Adjust Iterations	If the Train Action option is set to Search or Increment, and this option is set to Yes (default), the Maximum Iterations value is adjusted higher if the selected iteration equals the previously used Maximum Iterations value.
Freeze Connections	This option specifies whether connections (weights) should be frozen. The default setting for the Freeze Connections property is No.
Total Number of Hidden Units	This option specifies the hidden unit ceiling when Train Action is set to Search. Permissible values are 5, 10, 20, 30 (default), 40, 50, 75, or 100.
Final Training	This option indicates whether the final model should be trained again to allow the model to converge. If this is set to Yes, the number of iterations used will correspond to the value set in the Final Iterations property.
Final Iterations	This option indicates the number of iterations to use if the Final Training option is set to Yes. The Final Iterations property is unavailable if the Final Training option is set to No.

(c) Activation Functions

Activation functions alter and constrain a neuron's output range. By default, only the Direct, Normal, Sine, and Tanh activation functions are set to Yes (active).

Direct	Use the Direct activation function during training.
Exponential	Use the Exponential activation function during training if you want to constrain the output range of a neuron to nonnegative values ($0:\infty$)
Identity	Use the Identity (no transformation) activation function during training when you do not want to constrain the neurons activation range ($-\infty:\infty$).
Logistic	Use the Logistic activation (logit-link) function during training. The Logistic activation function returns the probability ($0:1$) of each target class.
Normal	Use the Normal (Gaussian) activation function during training. Its range is infinite ($-\infty:\infty$) although, in practical terms, it ranges from about -4.5 to 4.5.
Reciprocal	Use the Reciprocal activation function, ranging from $-\infty$ to ∞ , during training.
Sine	Use the Sine activation function during training. Its range is -1 to 1.
Softmax	Use the Softmax activation function (inverse generalized logit link). The Softmax activation function, described earlier, generates probability values ($0:1$) for each target class. It further ensures that the probabilities across the target classes sums to 1.
Square	Use the Square activation function during training. Like the exponential activation function, the square function (which squares its input) generates nonnegative values.
Tanh	Use the Hyperbolic Tangent activation function during training. The hyperbolic tangent is a sigmoidal activation function (like logistic) that ranges between -1 and 1.

The final section in the AutoNeural panel is the **Score** properties section, which enables you to specify the outputs to be generated. The available options are as follows:

Hidden Units	specifies whether you want to create hidden unit variables (default is No).
Residuals	specifies whether you want to create residual variables (default is Yes).
Standardization	specifies whether you want to create standardization variables (default is No).

SAS
THE POWER TO KNOW.

Available Train Options

You can choose from the following training actions:

- **Train:** Selected functions are trained until stopping.
- **Increment:** Nodes are added one at a time. It is retained only if a layer lowers the average error.
- **Search:** Nodes are added according to the specified architecture. The best network is retained. (This is the default.)

5

The *Train Action* option in the Train section of the AutoNeural properties panel specifies the method that you want to use to train the network. There are three options:


1. If **Train** is chosen, the specified hidden units are trained using all selected functions until stopping. There is no search. As a result, this option usually runs the fastest.
2. The **Increment** option adds nodes one at a time from each of the specified activation functions. If the added node lowers the average error, it is retained. If not, it is dropped. No function is reused.
3. When the **Search** (default) training action is selected, nodes are added according to the specified architecture. If there are hidden units, preliminary training is performed. Only the network generating the best fit performance is retained, where *best performance* is defined as per the following table:

Target Level	Validation Data?	Selection Criteria
Continuous	No	Schwarz-Bayesian Criterion (_SBC_)
	Yes	Validation Average Squared Error (_VAVERR_)
Discrete	No	Training Misclassification Rate (_MISC_)
	Yes	Validation Misclassification Rate (_VMISC_)

When **Search** is selected, the total number of hidden units is calculated on each run. If the value is greater than the specified (in the Total Number of Hidden Units property) number of hidden units, then training stops and the final model is selected. Due to the extra steps involved in conducting a search for the best performing network, the Search option usually runs the slowest.



Although AutoNeural's predictions are adjusted for priors, the model selection process is based only on the fit metric used, *without prior adjustment*. This could lead to unexpected results if the primary and secondary outcome proportions are not equal.



Tolerance Options

Tolerance options configure the extent of the preliminary search.

- **Low** requires no preliminary training.
- **Medium** performs preliminary training (default).
- **High** performs preliminary training and applies a convergence criteria (ABSCONV=0.001).

6

The Tolerance option is used to configure preliminary training.

Specifying **Low** for the Tolerance option turns off preliminary training altogether. Because preliminary training helps avoid the worst local minima, this option should probably be used sparingly.

A tolerance level of **Medium** (default) or **High** requests that preliminary training be performed. If **High** is specified, the algorithm also applies the $ABSCONV < 0.001$ convergence criteria.

© 2016 SAS Institute Inc. All rights reserved.
sas THE POWER TO KNOW.

Error Functions

- Normal (default for interval targets)
- Cauchy
- Logistic
- Huber
- Biweight
- Wave
- Gamma
- Poisson
- Bernoulli distribution
- Entropy
- MBernoulli (default for discrete targets)
- Multinomial
- Mentropy


7

In addition to fitting normally distributed targets, the AutoNeural node offers a rich set of error (deviance) functions that enable you to fit non-normal targets directly, ***without the need for target transformation.*** The available error functions are the same set as previously defined for the Neural Network node:

Normal	used with any type of target variable to predict the conditional mean. It is most often used with interval targets for which the noise distribution is approximately normal with constant variance. With categorical targets it is used for robust estimation of posterior probabilities.
Cauchy	used with any type of target variable. However, it is most often used when you want to predict the approximate conditional mode instead of the conditional mean.
Logistic	used with any type of target variable, but it is most often used with interval targets where the noise distribution might contain outliers.
Huber	suitable for unbounded interval targets that have outliers. It can also be used for categorical targets to predict the mode rather than the posterior probability.
Biweight	most often used with interval targets where the noise distribution might contain severe outliers although it can be used with any type of target variable.
Wave	used with any type of target variable, but it is most often used with interval targets where the noise distribution might contain severe outliers.
Gamma	most often used when the standard deviation of the noise is proportional to the mean of the target. It can be used only with positive interval target variables.
Poisson	suitable for skewed, nonnegative interval targets, especially counts of rare events, where the conditional variance is proportional to the conditional mean.
Bernoulli	only suitable for a target with the values 0 and 1.
Entropy	for independent interval targets with values between 0 and 1 inclusive.
Mbernoulli	suitable for all categorical targets, including multinomial.

- Multinomial can be used only with two or more interval target variables with nonnegative values, where each case represents a number of trials that are equal to the sum of the target values. The activation function must force the outputs to sum to 1 (for example, softmax).
- Mentropy should normally be used only with two or more interval targets with values that lie between 0 and 1 inclusive, and that sum to 1 for each case. The activation function must force the outputs to sum to 1 (for example, softmax).

For a categorical target, the default error function is multiple Bernoulli (MBernoulli). For interval targets, the default error function assumes that the target is normally distributed (Normal).



Termination Criteria

Training automatically terminates when the following occurs:


- **Overfitting** is detected (default).
- Maximum **training time** has elapsed.
- Reduction in **training error** is < 0.001 .

8

Training always stops when the maximum run time is exceeded. The default termination criterion is **Overfitting**, which, as its name implies, stops training when overfitting is detected.

Overfitting can be falsely detected for some models. Therefore, two additional options are available. If **Training Error** is specified, training will stop when the reduction in error is less than 0.001.

Specifying **Training Time** causes the models to be trained until the allotted time elapsed.




7.01 Poll

The AutoNeural node can build neural networks with more than one hidden layer.

- ☐ True
- ☐ False

9

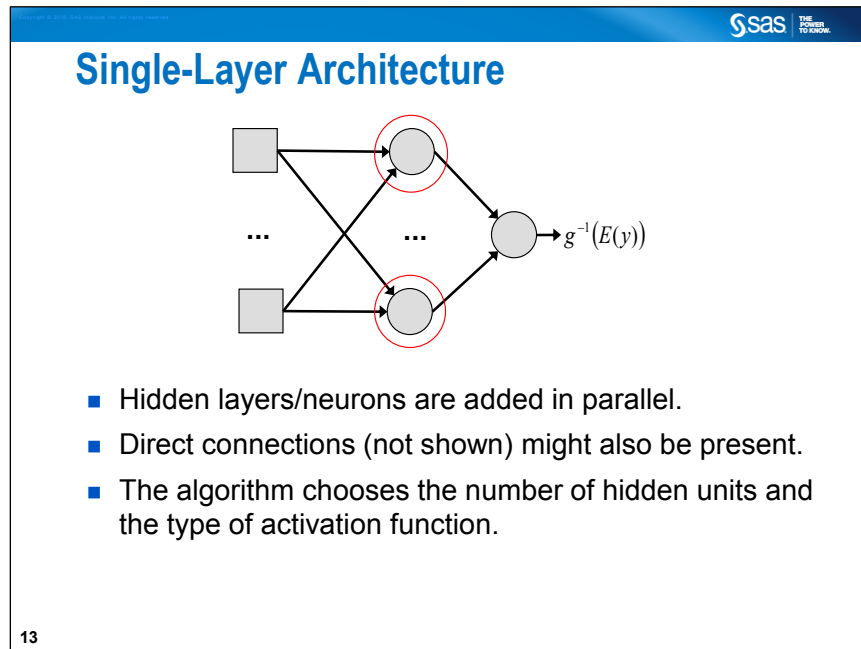
7.2 Architectures

sas
THE POWER TO KNOW

Objectives

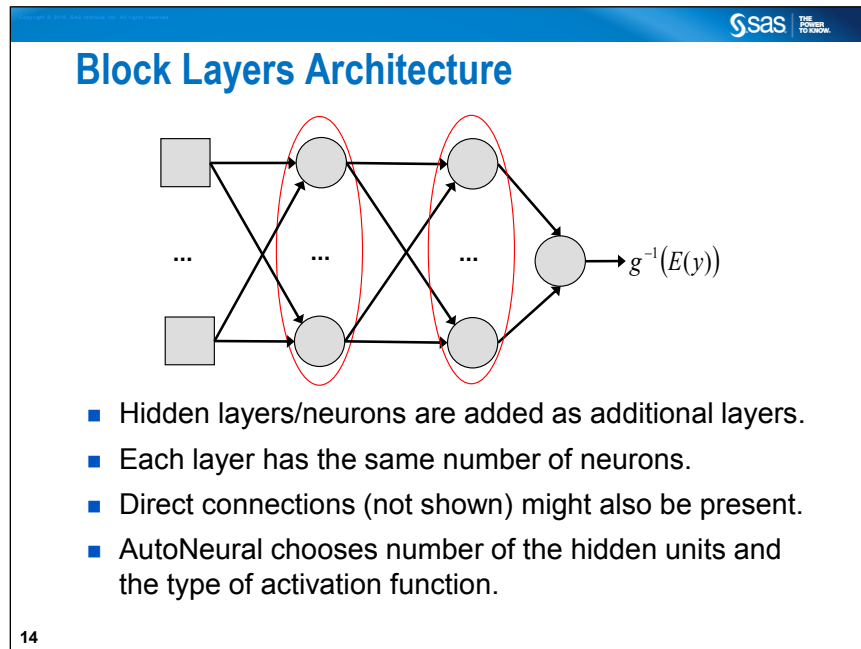
- Describe the four AutoNeural architectures:
 - Single Layer
 - Block Layers
 - Funnel Layers
 - Cascade

12



The Architecture property is used to specify the form of the neural network that you want to construct. Four predefined architectures are available. By default, a *single layer* architecture is created, in which ***the specified number of hidden units is added in parallel to the existing hidden layer.***

Setting the Number of Hidden Units property to **1** builds the network one neuron at a time. If the specified value is greater than 1, groups of neurons are added.



In the *block layers* architecture, **hidden units are added as additional layers**. Each layer that is added has exactly the same number of hidden units.

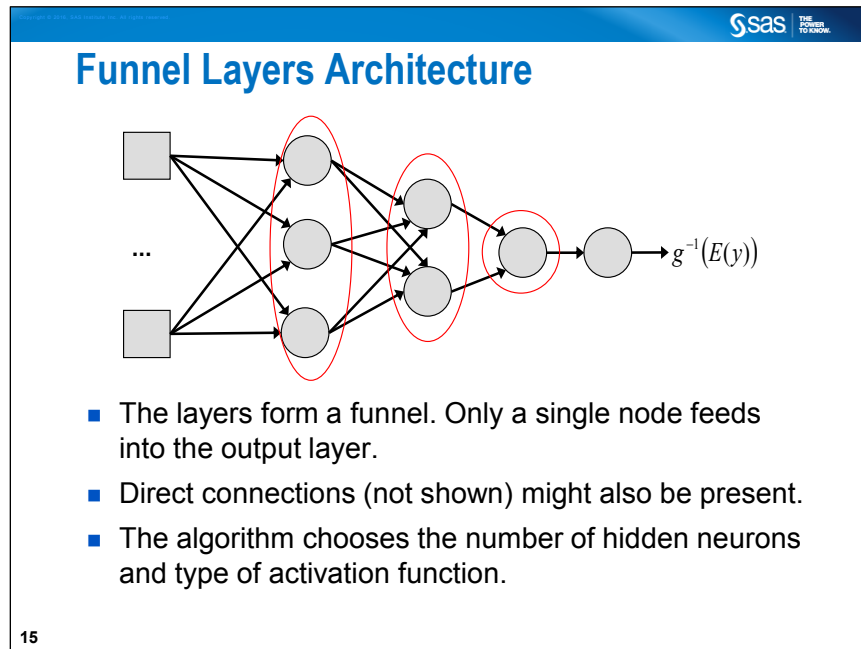


The number of hidden units added with each layer is given by the value specified for the Number of Hidden Units property.

Adding more hidden layers creates new levels of nesting, rather like combining separate MLP networks. Although a single-hidden-layer MLP models any continuous input-output relationship, additional hidden layers might be required when the relationship is discontinuous. Also, somewhat counterintuitively, the addition of an extra hidden layer can actually often help speed up convergence on a solution.

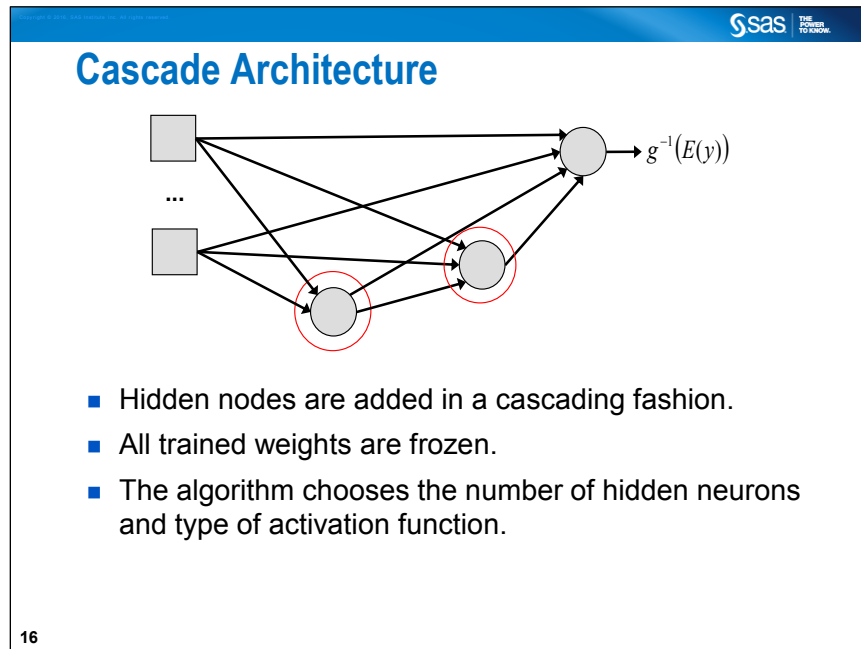
Despite these advantages, the block layers architecture is **not** commonly used. A key reason for its lack of popularity is that the number of weights grows quickly with the number of hidden layers. Specifically, assuming that there are k inputs, and that h_1 and h_2 are the number of units in the first and second hidden layers, respectively, then the number of weights that must be set is given by the following equation:

$$h_1(k+1) + h_2(h_1+1) + h_2 + 1$$



As in the block layers architecture, in the *funnel layers* architecture, multiple hidden layers are created. However, ***each time a new layer is added, a single neuron is added to the previous layer***. This forms a funnel pattern, and a single neuron feeds into the output layer. (See the diagram above.)

The funnel layers architecture is also rarely used.



The *cascade* architecture (Fahlman and Lebiere 1990, Prechelt 1997) is a multilayer perceptron in which **each hidden unit is connected to all preceding input and hidden units in a cascading fashion**, giving the algorithm its name. For example, the output of a two-input (x_1, x_2), two-hidden-unit (H_1, H_2) network that uses the hyperbolic tangent (\tanh) activation function is given by this equation:

$$g_0^{-1}(E(y)) = w_0 + w_1x_1 + w_2x_2 + w_3H_1 + w_4H_2$$

where

$$H_1 = \tanh(w_{01} + w_{11}x_1 + w_{21}x_2)$$


$$H_2 = \tanh(w_{02} + w_{12}x_1 + w_{22}x_2 + w_{32}H_1)$$

Notice that hidden unit H_1 is included in the input set to hidden unit H_2 . If a third hidden unit (H_3) is added, its activation would be given by this equation:

$$H_3 = \tanh(w_{03} + w_{13}x_1 + w_{23}x_2 + w_{32}H_1 + w_{43}H_2)$$

And the corresponding output equation would be as follows:

$$g_0^{-1}(E(y)) = w_0 + w_1x_1 + w_2x_2 + w_3H_1 + w_4H_2 + w_5H_3$$

 Regardless of the Number of Hidden Units value that is specified, **the Cascade architecture always constructs its networks one hidden unit at a time.**

Details

The Cascade network construction methodology is described below.

1. Fit a generalized linear model.
2. Add a hidden unit and connect it both to all input units and to all of the previous hidden units. This, in effect, turns all previous hidden units into inputs to the new hidden unit.
3. To initialize the new hidden unit to reasonable parameter values, freeze all of the weights and biases (except for the output layer biases) in the old network at their present value, and train.

4. Thaw the frozen parameters, allowing all of the weights to again change, and then re-train the entire network. This integrates the new hidden unit's weights with the rest of the network.
5. If convergence has not been achieved, then return to step 2.



Comparing AutoNeural Architectures

c7s2d1

This demonstration explores the performance of the AutoNeural node's single layer, block layers, funnel layers, and cascade (correlation) architectures. It compares the impact of changing the Train Options setting within each architecture from **Train** to **Increment**, and then to **Search**. The architectures are assessed against the **TITANIC** data table, described in Chapter 2.


1. Right-click **Diagrams** in the project panel and select **Create Diagram**.
2. Enter **autoneural** in the **Diagram Name** field, and click **OK**.
3. Because the **TITANIC** data source has already been defined, just drag and drop it onto the diagram.

Single Layer Architecture (Default)

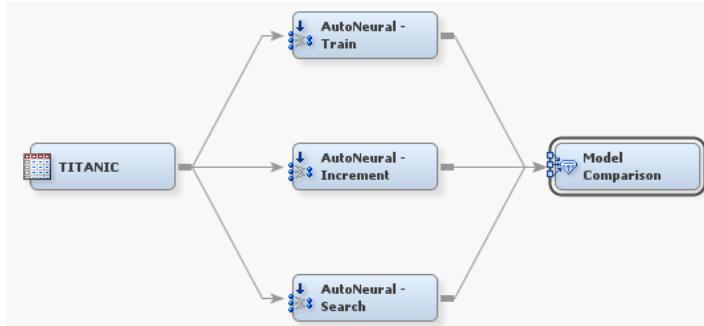
4. Add an **AutoNeural** node (leftmost node in the Model tab) to the diagram.
5. Connect it to the **TITANIC** data source.



6. Activate the AutoNeural node's properties panel, and make the following changes:
 - a. Change the Train Action property to **Train**.
 - b. Set the Maximum Iterations property (Model Options section) to **50**, which is the maximum allowed. This means that, in total, up to 50 groups of hidden units could be added.
 - c. Set the Number of Hidden Units property (Model Options section) to **1**. This means that "groups" of only one hidden unit will be added to the specified architecture on each iteration.

 AutoNeural always builds its cascade networks one hidden unit at a time, regardless of the value specified for the Number of Hidden Units property.
 - d. Change the Tolerance value (Model Options section) to **High**. This will not only perform preliminary training, it also applies the ABS CONV=0.001 convergence criteria.
 - e. Because the addition of a skip layer seldom helps improve a neural network's fit, under the Activation Functions section of the properties panel, set the Direct property to **No**.
 - f. In order to simplify the subsequent discussion, also set Sine to **No**. Only the Tanh (hyperbolic tangent sigmoid) and the Normal (Gaussian) activation functions will be used.
7. Right-click the **AutoNeural** node again and select **Rename** from the pop-up menu. Rename the node **AutoNeural - Train**.
8. Right-click the **AutoNeural** node and select **Copy** from the pop-up menu.
9. Right-click anywhere in the diagram white space and select **Paste** from the pop-up menu.
10. Connect the new node to the input data source.
11. Rename the copied node **AutoNeural - Increment**.
12. Change its Train Action value to **Increment**.
13. Add another copy by again right-clicking in the diagram white space and selecting **Paste**.
14. Connect the new node to the input data source.

15. Right-click the new node, and rename it **AutoNeural - Search**.
16. Change its Train Action value to **Search**.
17. Lastly, under the Assess tab, add a **Model Comparison** node to the diagram. The completed data flow diagram is shown below.



18. Run the diagram from the Model Comparison node, and open the results when asked.
19. Maximize the Output panel (lower right corner). The fit statistics appear at line 29.

Fit Statistics						
Model Selection based on Train: Misclassification Rate (_MISC_)						
Selected			Train:	Train:	Train:	
Model	Model Node	Model Description	Misclassification Rate	Average Squared Error	Roc Index	Train: Gini Coefficient
Y	AutoNeural3	AutoNeural - Search	0.19404	0.14523	0.839	0.678
	AutoNeural	AutoNeural - Train	0.20932	0.15426	0.819	0.637
	AutoNeural2	AutoNeural - Increment	0.21008	0.15503	0.815	0.630

The Search variant of the Single Layer architecture generated the lowest misclassification rate and average squared error. It also had the largest ROC Index and Gini coefficient.

20. Close the Multiple Comparison node Results window, and return to the diagram.

Block Layer Architecture

21. Activate the **AutoNeural - Train** node's properties panel, and change the Architecture property to **Block Layers**. *Make the same change in the other two AutoNeural nodes.*
22. Re-run the diagram from the Model Comparison node, and examine the results when they are ready.
23. Maximize the Output panel, and examine the performance information at line **29** (shown below).

Fit Statistics						
Model Selection based on Train: Misclassification Rate (_MISC_)						
Selected			Train:	Train:	Train:	
Model	Model Node	Model Description	Misclassification Rate	Average Squared Error	Roc Index	Train: Gini Coefficient
Y	AutoNeural3	AutoNeural - Search	0.22002	0.15961	0.798	0.596
	AutoNeural2	AutoNeural - Increment	0.22154	0.15966	0.798	0.596
	AutoNeural	AutoNeural - Train	0.22231	0.15968	0.798	0.596

When the block layers architecture is specified, the Search variant again outperformed both the Train and Increment options with respect to misclassification rate and average squared error. However, the Train option had the largest ROC Index and Gini Coefficient values. In all cases, the difference in the goodness-of-fit scores for the three training actions (that is, Train, Increment, and Search) was minimal

24. Close the results window and return to the diagram.

Funnel Layer Architecture

25. In the AutoNeural - Train node's properties panel, change the Architecture property to **Funnel Layers**. Make the same change in the other two AutoNeural nodes.
26. Re-run the diagram from the Model Comparison node, and examine the results when selection is complete. (The Search variant might take some time to finish.)
27. As before, maximize the Output panel and examine the fit statistics at line **29** (reproduced below).

Fit Statistics						
Model Selection based on Train: Misclassification Rate (_MISC_)						
Selected			Train:	Train:	Train:	
Gini			Misclassification	Average Squared	Roc	Train:
Model	Model Node	Model Description	Rate	Error	Index	Train: Coefficient
Y	AutoNeural3	AutoNeural - Search	0.19939	0.15155	0.825	0.651
	AutoNeural	AutoNeural - Train	0.20397	0.15088	0.825	0.649
	AutoNeural2	AutoNeural - Increment	0.20550	0.15239	0.811	0.623

Again the Search variant outperformed the Train and Increment variants on most of the measures. The Train did tie Search on the ROC Index, and it had the lowest average squared error of the three. As in the Block Layer Architecture, the difference in the fit of the three training actions was minimal.

28. Close the Results window and return to the diagram.

Cascade Architecture


29. Lastly, in the AutoNeural - Train node's properties panel, change the Architecture property to **Cascade**. Make the same change in the other two AutoNeural nodes.
30. Rerun the diagram from the Model Comparison node, and examine the results.
31. Maximize the Output panel and examine the fit statistics at line 29 (reproduced below).

Fit Statistics						
Model Selection based on Train: Misclassification Rate (_MISC_)						
Selected Model	Model Node	Model Description	Train:	Train:	Train:	Train: Gini Coefficient
			Misclassification Rate	Average Squared Error	Roc Index	
Y	AutoNeural3	AutoNeural - Search	0.15966	0.12135	0.889	0.777
	AutoNeural	AutoNeural - Train	0.20474	0.15229	0.823	0.645
	AutoNeural2	AutoNeural - Increment	0.22078	0.15784	0.810	0.619

Again the search option generated the best model, as defined by the smallest misclassification rate on the training data. In fact, in all four architectures, the search train option generated the smallest misclassification rate. And in three of the four architectures, it also has the smallest average squared error value, and the largest ROC index and Gini coefficients. Thus, the "Search" option would seem to be the best choice. The cascade architecture's Search variant had the smallest misclassification rate overall. It also had the smallest average squared error and the largest ROC index and Gini values.

32. Close the Results window and return to the diagram.
33. Close the AutoNeural node's Results window.
34. Close the "autoneural" diagram.

End of Demonstration

sas
THE POWER TO KNOW.

7.02 Multiple Choice Poll

Which of the following use multiple hidden layers?

- a. the block layers architecture
- b. the funnel layers architecture
- c. the cascade architecture
- d. all of the above

18

7.3 Chapter Summary

The AutoNeural node provides a convenient way to automatically determine an appropriate neural network architecture for a given data table. Four predefined architectures are available:


- By default a single (hidden) layer is created, in which the hidden units are added in parallel.
- If the block layer architecture is specified, hidden layers are added as additional layers. Thus, the block architecture enables you to construct multiple hidden-layer networks.
- In a funnel architecture, the output layer has only one unit, the preceding layer has two hidden units, the layer before that has three hidden units, and so on.
- In the cascade architecture, each new layer (a single hidden unit because units are always added one at a time) is connected to all preceding layers, including the input layer. This, in essence, turns each new hidden unit into an input unit on any subsequent hidden unit additions.

There are also three training actions from which you can choose.

- The default training action is **Train**. When **Train** is selected, all hidden units are trained using all selected activation functions until stopping. There is no search, so this option is usually the fastest.
- An alternative to the default training action is **Increment**. The Increment option adds nodes from each of the selected activation functions, one at a time. No activation function is reused. If the added layer lowers the average error, it is retained. If not, it is dropped from the final model.
- As its name implies, the **Search** option searches for the best network, adding nodes according to the specified architecture. It is the most computationally intensive option, particularly if you do not use a validation data set. When requested, preliminary training is performed.

7.4 Solutions

Solutions to Student Activities (Polls/Quizzes)




7.01 Poll – Correct Answer

The AutoNeural node can build neural networks with more than one hidden layer.

☒ True
☐ False

10



7.02 Multiple Choice Poll – Correct Answer

Which of the following use multiple hidden layers?

- a. the block layers architecture
- b. the funnel layers architecture
- c. the cascade architecture
- ☒ d. all of the above

19

Only the single-layer option is restricted to one hidden layer. Apart from building multiple hidden-layer networks using PROC NEURAL coding (as illustrated), currently the only way to build neural networks with more than one hidden layer is by means of the AutoNeural node.