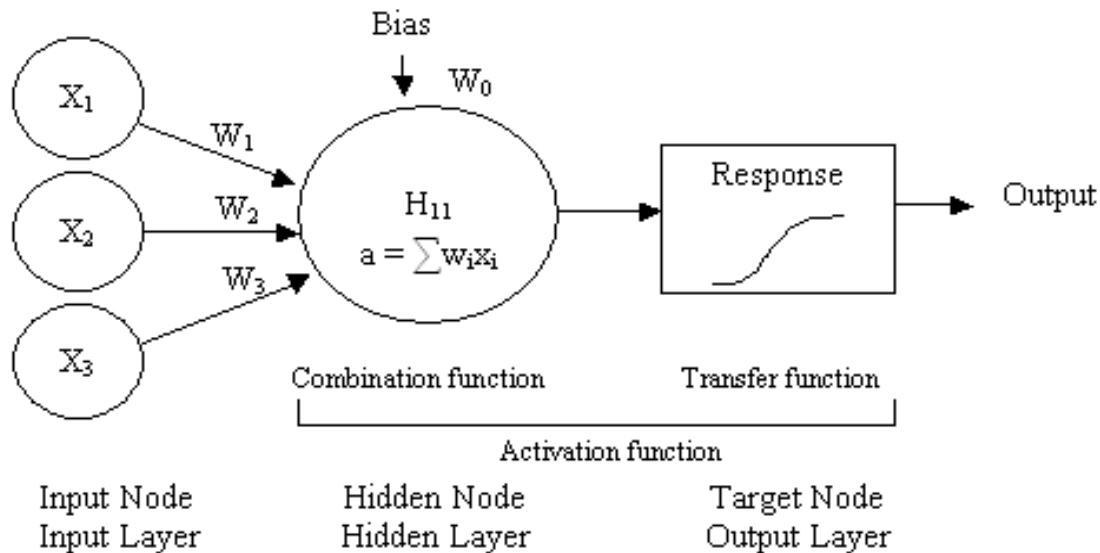


### 3.3 Multiple Layer Perceptron Architecture

Figure 3.3



*A feed-forward neural network MLP architecture*

#### ***An Overview to Neural Network Modeling***

Figure 3.3 displays the network diagram that represents the corresponding statistical model of a MLP multiple layer perceptron architecture consisting of one input layer with three input nodes, neurons or units ( $X_1$ ,  $X_2$ ,  $X_3$ ) with three input weights ( $W_1$ ,  $W_2$ ,  $W_3$ ) going into a single hidden layer with one hidden unit and an activation function that is connected to a single output unit.

The input layer consists of units for each *input* variable. The output layer of units are called the *target* variables. The hidden layer of the neural network design contains hidden units or neurons. The hidden layer determines the multiple layer network that applies a nonlinear transformation to the linear combination of input layer and hidden layer units to generate the *expected target* values or the predicted values. In a single layered network, each output receives inputs from all the input units, sums the units or neurons and mathematically transforms the sum typically using a nonlinear transfer function to calculate the predicted values called the expected target values.

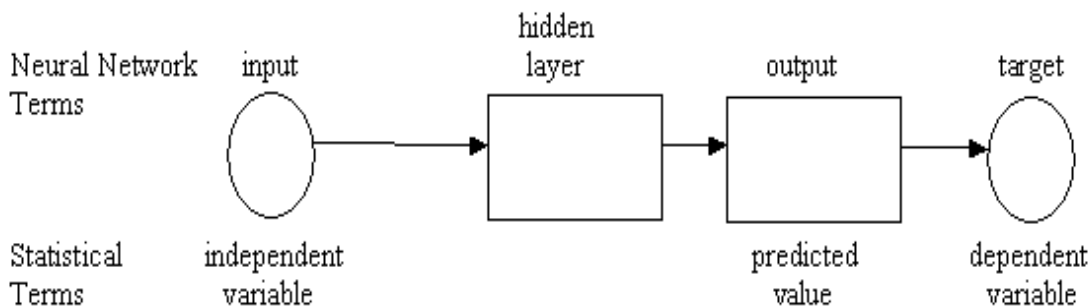
Neural network modeling is designed to predict the values of the *output* or response variable called the *expected target* values from other unknown *inputs* or predictor variables. The target variable may be continuous (linear regression) or categorical (logistic regression or discriminant analysis). That is, neural network modeling can either perform regression or classification modeling. The target variable level of measurements can be either *interval* i.e. continuous (e.g. (2, 23, 12, 33)), *ordinal* (discrete values with a logical ordering (1, 2, 3)), *nominal* (discrete values with no logical ordering (AZ, CA, NY)) or *binary* values which take two distinct values e.g. (Yes, No). The level of measurement of the input variables can also be interval, ordinal or nominal. However, numerical codes must be assigned to the categorical values to both the target and input variables in the predictive model.

**Type of Modeling Based on the Level of Measurement of the Target Variable**

Type of Target Variable	Measurement Scale	Type of Modeling
Continuous	Interval	Many different models
Independent	Interval	Regression modeling
Dependent	Interval	Time Series modeling
Dichotomous	Binary	Logistic Regression modeling
Multiple Discrete Responses	Nominal	Generalized Logistic modeling
	Ordinal	Cumulative Logistic modeling

In linear regression modeling, it's assumed that the target variable is linearly related to the input variables to the model. Neural network modeling assumes a nonlinear relationship between the target variable and the input variables in the model that is based on the architecture and the activation function applied to the network. Neural network modeling is capable of modeling a wide variety of extremely complex nonlinear functions and highly nonlinear decision surfaces. The *weights* represent the regression coefficients to the neural network model. It's important to select the most significant input layer and output layer weight estimates in the model. That is, similar to traditional regression modeling where we are looking for a high correlation or effects with the input variables in predicting the target variable. And a common technique called *network training* is applied where we fit the neural network model any number of times, usually between 10 to 50 times, and generating a different set of randomly generated starting values to the initial weight estimates that are set at some small numbers and then selecting the best neural network model based on the smallest modeling assessment statistics.

The neural network consists of units or neurons and connections between the units. There are three types of units. The *input units* represent the vector of inputs or independent variables where each input unit has its own weight. The input units are standardized assuming that they have an interval level of measurement. Standardizing the inputs is very effective when the variables are measured in different units. The input vectors or *input units* are used to predict the values of the output vector or target variable. The *hidden units* perform an internal nonlinear transformation and the *output units* generate the predicted values then compares the difference between the predicted values with the values of the output units that is called the *error*. *Connections* are used to pass information from units to other units.

**Figure 3.4**

A neural network may contain many units with the units grouped into layers. There can exist many input layers, hidden layers and output layers. The input layer is the first layer and the output layer is the last layer. The intermediate layers are called the hidden layers.

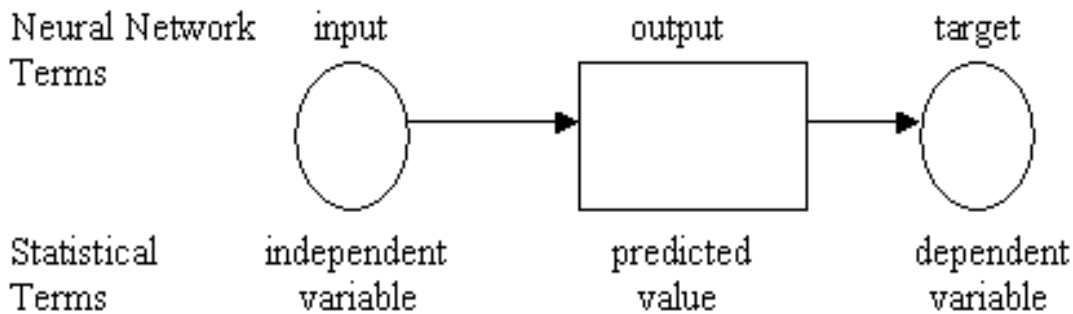
Connecting two layers in a neural network design, every unit in the first layer is connected to every unit in the second layer. But at times, units in the first layer can bypass one or more hidden layers and directly connect to the output layer called a *skip layer* that is shown in Figure 3.9. A skip layer connection is essentially traditional linear regression modeling in a neural network design. In time series modeling, a skip layer or a direct connection is applied to a MLP design in order to adjust for nonstationarity in the time series data over time. Each input unit is connected to each unit in the hidden layer and each hidden unit is connected to each output unit. The hidden units combine the input values and a transfer or activation function is applied. These hidden layers might have many different activation functions. The activation function applied to the output layer depends on the type of target variable and the values from the hidden units that are combined at the output units with additional activation functions applied that might be different. All units in a given layer have the same attributes. That is, all the units in the input layer share the same level of measurement and the same method of standardization. The hidden layer units share the same combination function and activation function. And all the units in the output layer share the same level of measurement, combination function, activation function and error function. An *activation function* is a mathematical transformation of the summarized weighted input units to create the output of the neuron. That is, the activation function is in two parts. The first part applies a *combination function* that accumulates all the input units into a single value. The combination function is usually a general-linear combination that is a weighted sum where each unit is multiplied by its associated weight and added together into a single value. The second part of the activation function is called the *transfer function* that usually applies a nonlinear transformation to these summarized units to generate an output unit. The type of transfer function applied in the hidden layer depends on the type of combination function that is applied. The selection of the correct transfer function to apply is not as important as specifying the appropriate type of activation function to apply.

By default, a *MLP multi-layer perceptron* is a feed-forward neural network architecture that uses various linear combination functions and nonlinear sigmoidal activation functions. A MLP architecture is a nonlinear regression model. But, there are other combination functions and activation functions to apply that connect the layers in many different ways i.e. the hidden layer might have many hidden units with different activation functions. But, only one type of activation function is allowed to each hidden layer. A second hidden layer could be added to the design. But, since a MLP architecture with one hidden layer can virtually forecast any model to any degree of accuracy assuming that we have an adequate number of hidden layer units, a sufficient amount of data and a reasonable amount of computational time. Therefore, additional hidden layers are usually not required. But, adding more hidden units to a neural network architecture gives the model the flexibility of fitting extremely complex nonlinear functions. This also holds true in classification modeling in approximating any nonlinear decision boundary with great precision. Again, by adding additional hidden units to a feed-forward neural network design is similar to adding additional polynomial terms to a polynomial model. This process is called *generalization*. Generalization is a process in choosing the appropriate complexity to the model in generating accurate prediction estimates based on data that is entirely separate from the data that was used in network training. That is, adding additional higher-order modeling terms to the nonlinear polynomial regression model will always lead to a perfect fit. However, a perfect fit may also lead to the danger of extrapolation and generating absurd predictions from the polynomial regression model having an entirely different functional form based on fitting the regression model beyond the range of the actual data points. This same analogy holds true in neural network modeling where adding too many hidden layer units might lead to the global minimum and poor generalization error, i.e. an undesirable test error, based on data that is entirely separate from network training. There are three conditions for good generalization. First, the input variables accurately describe or predict the output responses. Secondly, the function that we are want to fit is approximately smooth with small changes in the inputs resulting in small changes in the target values. And lastly, a sufficiently large training data.

### 3.4 An Explanation of the Neural Network Layers

#### The Simplest Neural Network Design

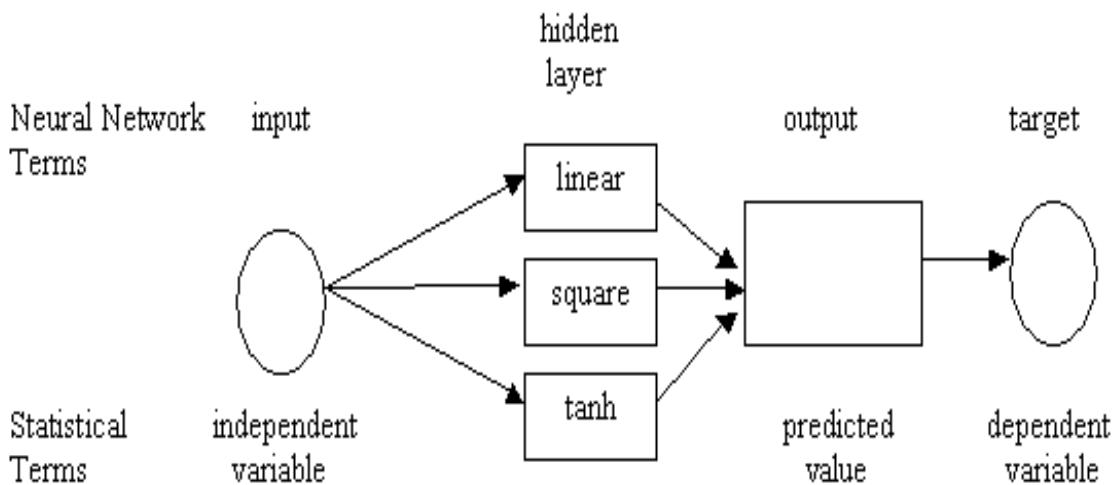
Figure 3.5



In traditional regression, the diagram represents a simple linear regression model with one independent variable and one dependent variable. In neural network terms, the network diagram represents the simplest network design with a single input unit (independent variable), a single target variable (dependent) and a single output variable (predicted variable).

A neural network may consist of numerous units. The units are combined and connected into layers. There can exist several input layers, hidden layers and output layers. The basic configuration in connecting two separate layers in a neural network design is by connecting every unit in the first layer to every unit in the second layer. The input units from the input layer are standardized in order for the variables to have the same unit of measurement by using the same method of standardization. Again, all the units in the hidden layer have the same combination function and activation function and all units in the output layer have the same combination function, activation function and error function.

Figure 3.6



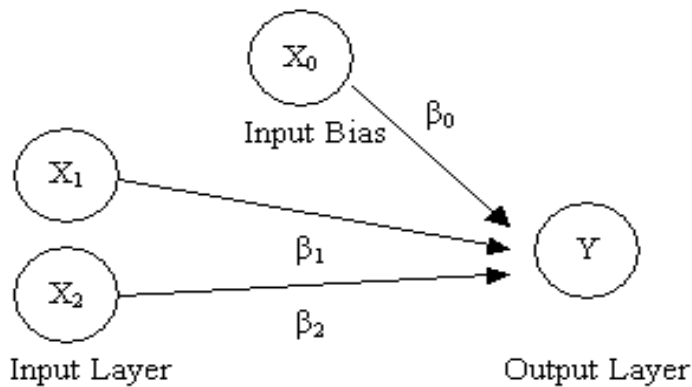
*A MLP neural network design with three hidden layers and a different activation function for each hidden layer unit.*

### 3.5 Relationship Between the Predictive Models

The following section in explaining the relationship between multiple regression parameter estimates and the neural network weight estimates is taken from an excerpt from “Applied Linear Statistical Models”, 5<sup>th</sup> Edition by Kutner, Nachtsheim, Neter and Li (2004).

#### Multiple Linear Regression Model

Figure 3.7



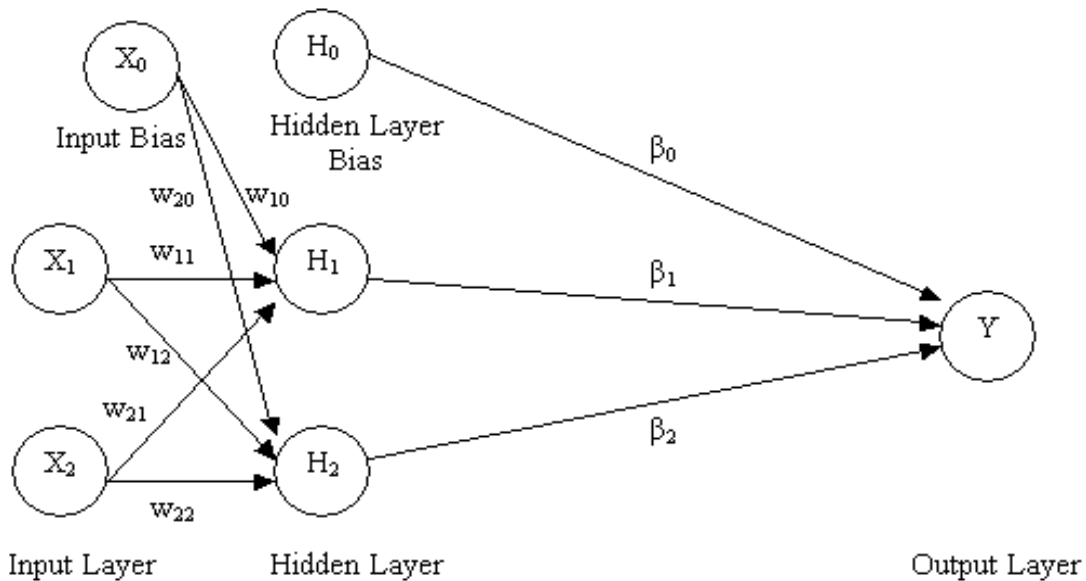
A multiple linear regression model with two input variables and an intercept term to the model.

#### Multiple Linear Regression model

$$E(Y) = \beta_0 + \beta_1 \cdot X_1 + \beta_2 \cdot X_2$$

#### Feed-Forward Neural Network Model

Figure 3.8



A feed-forward neural network model with two input units, a single hidden layer with two hidden units and biases and a hidden-to-output layer bias applying a linear activation function.

### Feed-Forward Neural Network Model

$$E(Y) = \beta_0 + \beta_1 H_1 + \beta_2 H_2 \quad \text{where } H_1 = w_{10} + w_{11}X_1 + w_{21}X_2 \text{ and} \\ H_2 = w_{20} + w_{12}X_1 + w_{22}X_2$$

$$E(Y) = \beta_0 + [\beta_1 w_{10} + \beta_1 w_{11} + \beta_1 w_{21}] + [\beta_2 w_{20} + \beta_2 w_{12} + \beta_2 w_{22}] \\ + [\beta_1 w_{11} X_1 + \beta_1 w_{21} X_2] + [\beta_2 w_{12} X_1 + \beta_2 w_{22} X_2] \\ = b_0 + b_1 X_1 + b_2 X_2$$

$$\text{where } b_0 = \beta_0 + [\beta_1 w_{10} + \beta_1 w_{11} + \beta_1 w_{21}] + [\beta_2 w_{20} + \beta_2 w_{12} + \beta_2 w_{22}] \\ b_1 = \beta_1 w_{11} + \beta_1 w_{21} \\ b_2 = \beta_2 w_{12} + \beta_2 w_{22}$$

Neural network modeling is essentially multiple linear regression assuming that a linear activation function is applied to the hidden layer. Initially, network training is performed by fitting the neural network model as an intercept-only model that perfectly predicts the average target values.

### Comparing the Number of Parameters Between Both Predictive Models

The interpretation of the neural network parameter estimates are extremely difficult to interpret even when there is no transformation applied to the hidden layer that is indicated from the above formulas. The reason is because the weight estimates are a function of both the input-to-hidden layer and the hidden-to-target layer weight estimates. And adding additional input variables and hidden layer units to the neural network model will also dramatically increase the complexity to the neural network design with an exponential increase in the dimensionality of the nonlinear error function. Thereby, making it nearly impossible in locating a desirable minimum to the nonlinear error function. Conversely, adding interaction terms and polynomial terms to the regression model will also exponentially increase the complexity to the predictive model.

The neural network design increases in complexity based on the general formula. That is, the number of parameters to a MLP design with one hidden layer is defined as follows.

$$\text{number of parameters} = h(k + 1) + h + 1$$

where  $k$  is the number of input variables and  $h$  is the number of hidden units in the network model.

The number of hidden layer units applied to a MLP neural network model with a single input variable in the model plays a similar role in adding additional interaction or higher-order modeling terms to a least-squares regression model. That is, achieving a perfect fit to the model by adding enough hidden units to the neural network model or including enough interaction or higher-order modeling terms to the multiple linear regression model. However, fitting a model with absolutely no error will result in overfitting to the data and usually results in poor generalization. Selecting the correct number of hidden units to the neural network design is one of the most important decisions in producing excellent generalization, which is the main goal in neural network modeling.

### Comparing the Parameter Estimates Between Both Predictive Models

One strategy in comparing the neural network weight estimates with traditional regression estimates is by observing both the input layer weight estimates and the output layer bias term from the neural network model. That is, observing the size, order of magnitude and the sign of the neural network input layer weight estimates with comparison to the regression coefficients of the predictor variables and also observing the magnitude and sign of the hidden-to-target layer bias with comparison to the intercept term in the regression model. This will be introduced in the modeling comparison examples presented later in the book.

## 3.6 An Overview of the Neural Network Layers

### *Input Layer*

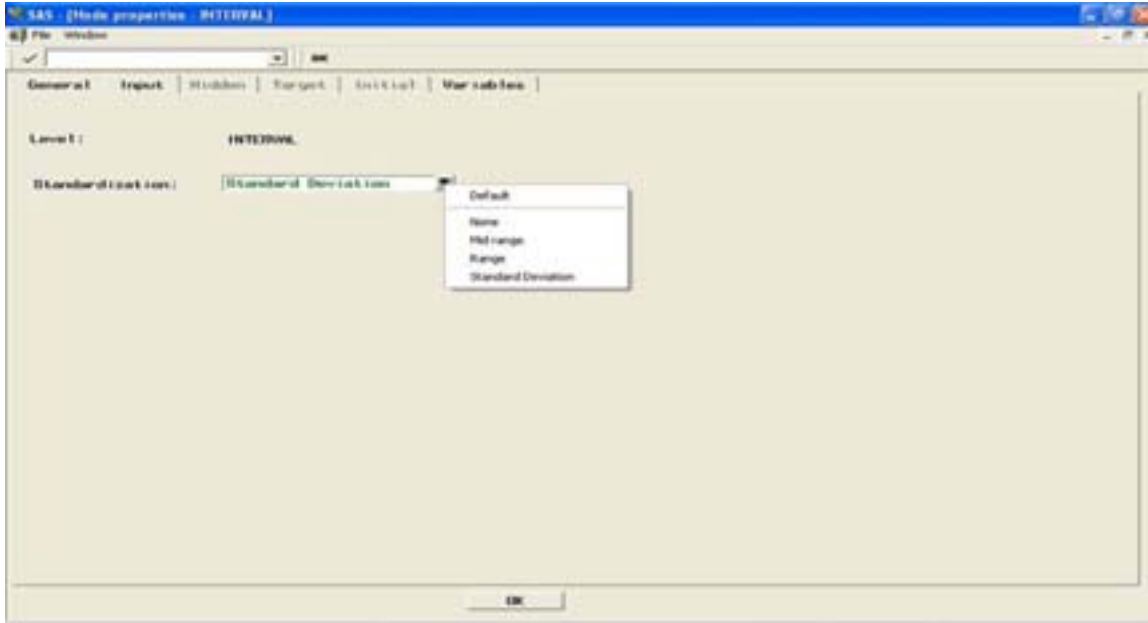
The *input layer* is the first layer to the neural network design. The input layer does nothing but represent the input variables to the network model. Enterprise Miner **Neural Network** node automatically creates one input layer node for interval, nominal and ordinal variables. That is, all the units in the input layer share the same level of measurement and the same method of standardization. Additional input layers can be created and variables can move from one layer to another.

For nominal-valued inputs with  $k$  levels,  $k-1$  input units will be automatically created. Therefore, at times the number of input units in the neural network design may be greater than the number of input variables. By default, the interval-valued input variables in the input layer are standardized by a standard normal distribution with the target variables in the hidden-to-target layer left unstandardized in the MLP or RBF design. Yet, standardization of the target variable assures convergence. The method of standardization can be specified for each input unit. Standardization can only be applied to continuous or interval-valued variables. The purpose of standardizing the input units is to produce reasonable values of the input-to-hidden layer weights close to zero to assure convergence, to reasonably compare the size of the weight estimates among the various input variables in the neural network model and to select a meaningful range to the initial input layered weight estimates. Standardization is very effective when the input variables in the model are measured in different units in order to interpret the variables to a common scale or compare the variables to a common unit. The following are the various standardization methods that can be applied to the interval-valued input layer units in Enterprise Miner.

#### *Standardization Methods Applied to the Input Units*

- **NONE:** Prevents standardizing the interval-valued input variables that is the default for categorical input variables in the network model or GLIM architectures.
- **MIDRANGE:** Subtracts the midrange and divides by half the range resulting in values with a maximum of 1 and a minimum of  $-1$ .
- **STANDARD DEVIATION:** This is the default standardization method. The standard deviation method subtracts the mean and divides by the standard deviation that transforms the input variable to a standard normal distribution with a mean of 0 and a standard deviation of 1. That is, assuming the input variable has a normal distribution about its mean.

Other reasons for standardizing the input variables in the neural network model is to reduce the risk of the iterative process resulting in a bad local minimum of the error function, accelerates neural network training, alleviates the “curse of dimensionality” and ill-conditioning. However, standardization should not be used if there doesn’t exist any bad local minimums in the error function. Also, the drawbacks to standardization is that it discards information in the variables and standardization is not recommended if there is a wide discrepancy in the scale of measurement of the variables. That is, the *scale of measurement* is the measurements of some attribute set to the corresponding variable e.g. pounds, dollars, Fahrenheit and so on. If the input variables are not standardized then other termination criterion under the neural network **Advanced** tab may need to be set. Also, standardization is applied is because the weight estimates cannot be compared to one another under the assumption that each input variable is measured entirely different. Applying a skip layer design with the input variables and the target variable in the neural network model left unstandardized will result in exactly the same parameter estimates and prediction estimates with comparison to a linear regression model. This type of neural network configuration is called a *Generalized Linear Model* architecture.



Selecting the standardization method to the input variables in the neural network model from the *Neural Network* node.

Select the **Basic tab** > **Advance user interface** > **Advanced tab**  
> **Double-Click Inputs** icon > **Input tab**

It's important to understand as each input variable is entered into the nonlinear neural network model then it will result in an additional dimension to the space in which the range of the data resides. Therefore, it's important that input variables are eliminated from the modeling design that provides very little information in describing or predicting the target values that will lead to an increase in the modeling performance and reducing the “curse of dimensionality” dilemma. Both the complexity of the neural network model, the number of internal calculations called function calls and computational time increases with additional parameters in the neural network model that is including additional input units, hidden layers and hidden layer units to the network design. This is especially true based on extremely large training data sets in which the nonlinear model is trying to fit since the iterative grid search procedure must make several passes through the entire data set in order to determine the next best set of estimates at each iteration. Increasing the number of inputs to the model will reduce the chance in determining the correct starting values to the weight estimates. Also, increasing the number of input variables to the model will also reduce the chance of the iterative grid search procedure from terminating too early. That is, increasing the number of input variables in the model will result in the optimization process inheriting the problem in deciding when the iterative grid search procedure should stop in determining the best linear combination of weight estimates. The reason is because the minimization technique will have problems in determining the minimum based on the increase in dimensionality in the neural network model. This is overcome by constructing neural network models with fewer input variables in the nonlinear model and the reduction in the dimensionality of the error space in determining the optimum minimum in the sum-of-squares error function.

The **Princomp/Dmneural** node and performing dmneural network training is specifically designed to overcome the “curse of dimensionality” phenomenon. Initially the dmneural network training procedure performs principal component analysis to the training data set in determining the best set of principal components based on the input variables in the predictive model. The best set of input variables that best explains the variability in the target responses is determined by the r-square

modeling selection criterion that we want as large as possible. Based on the iterative grid search procedure used in network training, the algorithm then determines a set of grid points from the selected principal component and a multidimensional frequency table from the training data set. The frequency table consists of frequency counts of the selected principal components at a specified number of discrete grid points. An activation function is applied to the linear combination of input variables that best explains the variability in the target values. That is, various activation functions are applied to the linear combination of input variables from the training data set at each stage of the neural network training algorithm that best explains the target values. However, keep in mind although principal components has its advantages, the drawback to this data reduction technique is that the highest principal component might not necessarily be the best component in explaining the variability in the target responses. Also, throwing-out the smallest components with smaller eigenvalues might actually remove important information in predicting the target responses.

Saturated neural network models will generate poor modeling performance and poor generalization with an increase in computational time that leads to overfitting. And a neural network model with no inputs and just an output bias ignores the oscillating behavior of the nonlinear function. Also, it's important to remember that excluding input variables that are critical in predicting the output responses will contribute to poor generalization. Therefore, the selection of the input variables is a critical part of neural network designing. However, at times the neural network models are known to consist of hundreds or even thousands of input variables to the neural network model depending on the size of the training data set. Determining the appropriate set of input variables in the neural network model may be determined by the various modeling selection techniques. Stepwise regression is the most popular modeling selection technique, even in neural network modeling. That is, it's important to determine the proper balance in the number of input variables in the predictive model that accurately predict the output responses that leads to good generalization.

It will usually take common knowledge to know which input variables to include in the neural network model that best explains the output responses. That is, specifying the correct specification of the target range and error distribution will increase the chance in selecting good inputs that should result in the best prediction estimates. Therefore, a simple rule of thumb is that if we have some general idea that the input variable will not be very useful in predicting the output responses then don't include it in the neural network model. That is, reducing the number of input units to the design will also reduce the number of hidden layer units connected to them. Therefore, resulting in better generalization based on the network training more efficiently. However, if it's important to include numerous input variables to the neural network model then it is critical that a MLP design is applied as opposed to some of the other RBF designs. The reason is because the MLP design is better at ignoring irrelevant input variables to the network model with comparison to some of the other RBF designs. This will be discussed shortly.

By default, the predicted values are automatically calculated for all cases with non-missing target values. However in the various modeling nodes in Enterprise Miner and the corresponding data-mining predictive modeling procedures, when the training data consists of missing values in any one of the input variables in the neural network model then these same cases are not used in training. Therefore, these same cases will not be scored. For interval-valued target variables, the predicted values of missing cases in the target variable are estimated by it's own mean. That is, the expected target values are set to an intercept-only prediction model. For categorical-valued target variables, the expected target values for each class level is estimated by the corresponding prior probabilities. Otherwise, the proportional frequencies of each class level in the training data is used. However, if the training data consists of numerous missing values in the input variables then SAS recommends using the **Replacement** node to impute missing values before fitting the predictive model then using the **Regression** or the **Neural Network** modeling nodes.

## Hidden Layers

In a layered neural network design, the *hidden layer* is an intermediate layer between the input and output layer where one or more activation functions are applied. The hidden layer is also called the *inner hidden layer* or the *input-to-hidden layer*. In neural networks, the hidden layers may apply additional transformations to the general-linear combination of input weight vectors. Generally, each input unit is connected to each unit in the hidden layer and each hidden unit is connected to each output layer unit. An activation function is applied that is usually nonlinear which calculates the units of the output by multiplying the values of each input unit and the hidden layer weight that are added up and then transformed using a sigmoidal function. The sigmoidal activation function is applied to the neural network design so that the output from the hidden units are close to zero with reasonable values of the hidden layer weights that are centered about zero. However, since a nonlinear function is applied to the hidden layer, therefore various optimization techniques must be applied in determining the weight estimates to the design. The more hidden neurons or hidden units added to the neural network architecture gives the model the ability in fitting extremely complex nonlinear smoothing functions such as polynomial functions that have a high-degree of oscillating behavior. The simplest network model consists of two hidden units since a network model with one hidden unit is simply a generalized linear model. It's not uncommon to have multiple hidden layers, but it is usually not recommended. Each hidden layer is connected to the following hidden layer until the last hidden layer is connected to the output layer. A single hidden layer is restricted to one type of activation function. Therefore, several hidden layers with different activation functions in each hidden layer can be applied that is illustrated in Figure 3.6. Applying a linear activation function to the hidden layer is essentially a neural network model with no hidden units called a single-layer perceptron displayed in Figure 3.1. Usually a single hidden layer unit is applied to a single input variable in the neural network model. But by adding more hidden units to the model, increases the complexity to the network design and can approximate any relatively smooth nonlinear function to any degree of accuracy. One of the biggest decisions in network designing is the number of units in the hidden layer. Selecting the correct number of hidden units is an important aspect in producing good generalization performance, which is the main goal in network training. The best number of hidden units to apply to the neural network design depends on the number of input variables to the network model, the number of observations in the training data and the noise level in the underlying distribution of the training data. Since, we do not know the complexity of the underlying training data or the random error in the output responses. Therefore, it is recommended to fit the network model numerous times with a different number of hidden units and analyzing the various modeling assessment statistics and stop the iterative process when the goodness-of-fit statistics begins to increase. The simplest approach is to initially fit the network model with no hidden layer units and sequentially fit the network model by adding an additional hidden unit each time. However, there isn't a general rule of thumb to tell you how many hidden layer units to apply to the network model.

If the neural network model has too few of hidden units it will lead to *underfitting*. Underfitting is basically a poor fit to the data. And conversely, fitting too many hidden units will result in overfitting. Overfitting is creating a model with too many parameters that does not generalize well. That is, increasing the complexity to the model that results in absurd predictions beyond the range of the actual target values. Generally, it's better having too many units in the hidden layer than too few. That is, in order for the network model to fit the nonlinear pattern in the underlying data or creating highly nonlinear decision boundaries for classification modeling a sufficient number of hidden layer units are needed. Bypassing the hidden layer and making a direct connection from the input layer to the output layer is called a skip layer design that is essentially least-squares regression modeling in a neural network design. Conversely, increasing the number of hidden layers in the neural network model will increase the accuracy in the estimates, but it will introduce overfitting to the data and can lead to ill-conditioning in the Hessian matrix. Therefore, one hidden layer is usually sufficient.

### Properties of the Hidden Layers

- **Combination function:** Generally used in a MLP architecture. Radial combination functions can be used for radial basis function networks to model nonlinear functions. The basic difference between both configurations is the way in which both architectures combine values from previous network layers. The MLP design uses the inner products of the input variables and the hidden layer weights while the RBF design applies a Euclidean distance between the input vector and the weight vector then multiplying by the bias term squared.
- **Activation function:** The default activation function applied to the hidden layer depends on the type of combination function and the number of hidden units. The default hidden layer activation function is the hyperbolic tangent function since it has a faster rate of convergence with comparison to the other sigmoidal functions assuming that we have a large sample size. However, the Elliott activation function can often produced similar estimates. Gaussian function can be useful for bumpy error surfaces with numerous local minimums. In RBF designs, the exponential activation function is applied in ORBF designs and a softmax activation function is applied in NRBF designs.
- **Bias term included to the model:** The bias term should almost always be included in each hidden layer unless the softmax activation function is applied for RBF architectures. A large hidden layer bias can reduce the complexity to the network model by causing redundant hidden units to be ignored. Also, there is one hidden layer bias for every hidden layer unit in the design.
- **Initial hidden layer bias has a random distribution:** By default, the initial hidden layer bias is set to a random normal deviate centered at a mean of zero with a small variability.

The following displays the relationship between the number of hidden layers that should be applied to the network design that is based on the type of activation function and combination function used in the neural network model that depends on the level of measurement of the target variable.

Type of Combination Function	# of Hidden Units	Default Activation Function
Additive	any	Identity
Linear	any	Tanh
Radial	one	Exp
	two or more	Softmax

### MLP Design with Two Hidden Layers

$$g^{-1}(E(y)) = w_0 + w_1H_1 + w_2H_2$$

where  $H_1 = g_1(w_{01} + w_{11}H_{11} + w_{12}H_{12})$  and  $H_2 = g_2(w_{02} + w_{21}H_{21} + w_{22}H_{22})$

where  $H_{i1} = g_i(w_{0i1} + w_{1i1}x_1 + w_{2i2}x_{12})$  for a given activation function  $g_i$  i.e.  $i = 1, 2$  with two input units, two hidden units and some activation function  $g_i(x)$

A second hidden layer can be applied to the neural network design that consists of a linear combination of weight vectors of the linear combination of the input weight vectors that are transformed in a nonlinear way. Computationally, the combination of separate MLPs generates the output. But, since a MLP model with a single hidden layer is called a *universal approximator*, therefore adding any additional hidden layers to the network model is not needed in estimating continuous nonlinear smoothing functions except in extremely rare circumstances in fitting extremely complicated target functions or reducing the number of weights to the neural network design. A universal approximation means that given a sufficient number of hidden layer units the neural network is designed to approximate any functional form to any degree of accuracy. However, this level of accuracy in a MLP design may not be achieved since the network weights and biases need to be estimated from the data and at times requiring an enormous number of hidden layer units.

## ***Output Layer***

The *output layer* is the final layer to the neural network design that represents the target variables to the model. The output layer is also called the *outer hidden layer* or the *hidden-to-target layer*. Usually, the output layer is fully connected to the hidden layer. The transfer function, combination function and error function that is applied to the output layer depends on the level of measurement of the target variable. All the units in a given output layer have the same level of measurement, combination function, activation function and error function. For interval-valued target variables, a unique output unit is created in the network design. For categorical target variables with  $k$  class levels,  $k - 1$  output units are created in the network design. The predicted values or the expected targets from the neural network model are calculated from the output layer. In the back-propagation technique, the error values that is the difference between the output responses and predicted values are calculated in the output layer that are then passed backwards, i.e. backpropagated, through the network to calculate the network error. Also for back-propagation, the derivative of the error function with respect to the weight estimates are also calculated from this layer that are then passed backwards through the network design in order to adjust the weights estimates and minimize the error in network training. For multivariate models, Enterprise Miner **Neural Network** node automatically creates one output layer unit for each categorical-valued target variable and one output layer unit for each interval-valued target variable in the model. That is, the output layer may have more than one output layer unit to the neural network modeling design that is analogous to multivariate regression modeling with multiple target variables to predict in the predictive model.

### ***Properties of the Output Layer***

- **Standardization:** By default, the target variables are not standardized. Only interval-valued target variables can be standardized. The target variables cannot be standardized applying a Poisson, binomial, multinomial or multiple entropy error function. And target variables are left unstandardized using a gamma, Bernoulli, entropy or multiple Bernoulli error function. The reason that the target variables are not standardized is because standardizing the target values makes it very difficult to interpret the responses. Although, standardization of the target variable helps to assure convergence to the iterative process. Also, standardization of the input and the target variables helps alleviate ill-conditioning with a large number of inputs or hidden layer units in the design. The purpose of standardization is to treat the target values equally. However, the drawback in standardizing the target variable is that its values must be remapped to their original scale. The reason that the target values need to be re-transformed to its original values is because the standardized values are usually not as informative as the original values.

### ***Standardization Methods Useful to the Output Units***

- **NONE:** Prevents standardizing interval-valued target variable that is the default method used in the output layer.
- **MIDRANGE:** Subtracts the midrange and divides by half the range resulting in the target values with a maximum and minimum of 1 and  $-1$  that is usually applied to tanh, arctangent, Elliott, sine and cosine activation functions.
- **RANGE:** Subtract the minimum and divides by the range that is usually applied to a nonnegative target variable with a log link function that has a minimum of 0 and a maximum of 1. That is, where the target values consists of rates or proportions. This type of standardization is usually applied to logistic, Gaussian and softmax activation functions.
- **STANDARD DEVIATION:** Standardizes the target variable into a standard normal distribution with a mean of zero and a variance of one assuming that the target variable is normally distributed about its mean.

- **Combination function:** Linear combination functions are generally applied to the output layers, except for ordinal target values where the EQSlopes combination function is applied. The type of combination function to apply depends on the level of measurement of the target.
- **Activation function:** The activation function applied to the hidden-to-target layer depends on the type of combination function that is applied and the level of measurement of the target variable. It's worth noting that the identity output activation function is applied for interval-valued target variables for computational convenience in network training. However, it's possible and often more desirable to consider other types of output activation functions that follows the distribution of the target values in both the RBF and MLP designs. The reason is that the distribution of the target values will force the data to conform to the output activation function. If it's possible, the target variable needs to be retransformed to their original values. The activation function that is applied to the hidden-to-target layer are the following:

Interval: Identity link function or no transformation

Ordinal: Logistic transformation

Nominal (Binary): Softmax transfer function i.e. a multiple logistic function

Nominal-valued targets having an error function of either a multiple Bernoulli, multiple entropy or multinomial distribution, then the only output transfer function that can be applied is the softmax activation function which is a multiple logistic function.

Ordinal-valued targets use only monotone increasing output transfer functions such as the identity, exponential, logistic, Elliott, tanh and arctan functions that are allowed.

**Note:** Selecting the appropriate output activation function corresponds to the distribution or the range of values of the target variable even after standardization is applied.

- **Bias term:** By default, the output bias is set to the target mean that is transformed by the output layer activation function. A bias term should almost always be included to each output layer unit unless the softmax activation function is applied in the NRBF design where the bias term is redundant and is omitted from the model. That is, the hidden-to-target bias would be absorbed or added to each weight estimate in the model. The value of the output layer bias can have a much greater magnitude than the other weights in the model given that the targets are not standardized. But, the modeling performance is does not depend on the bias term.
- **Error function:** The error function is designed to measure the accuracy of the modeling fit. For interval-valued targets, the error function is the sum-of-squares error function that we want at a minimum. For categorical-valued targets, the entropy error function is applied. The type of error function to apply depends on the level of measurement, the number of target variables in the output layer, the range of values of the target variables and the objective function.

Interval: Normal distribution

Ordinal: Multiple Bernoulli distribution

Nominal: Multiple Bernoulli distribution

**Note:** Usually the selection in the type of error function to use in network modeling depends on the distribution between the residual values and the fitted values. If the selection of the error function does not exist then you may want to transform the target responses so that the distribution of the residuals conforms to the error function that you have selected in the model.

- **Initial weight estimates:** By default, the starting values to the hidden-to-target layer weight estimates are set to zero. Hence, at the first iteration the hidden-to-target layer weights are set to the mean from the output activation function that can either move positively or negatively.

**Scale parameter:** The scale parameter is initialized to the standard deviation of the target variable when we randomize the initial target weight estimates.