

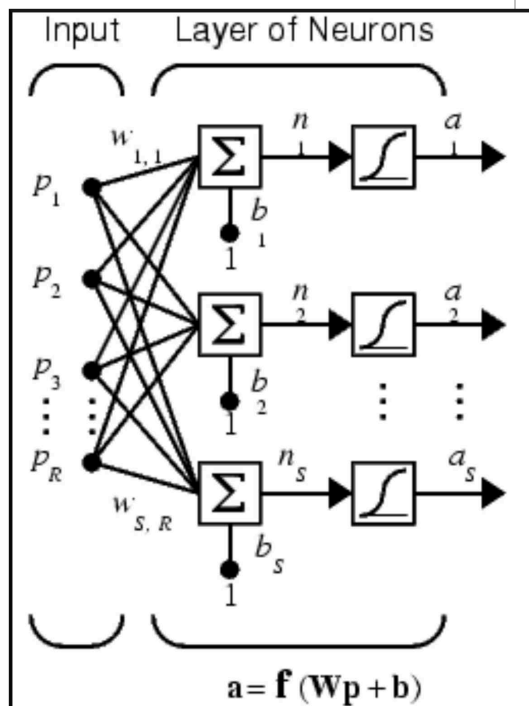
# DADiSP / NeuralNet

## Neural Network Module

DADiSP/NeuralNet is an add-on module to DADiSP that provides direct and easy access to the demonstrated predictive power and pattern recognition capability of neural networking technology. With DADiSP/NeuralNet, users can build their own artificial neural networks (ANNs) and apply them to achieve more accurate predictions and pattern classifications.

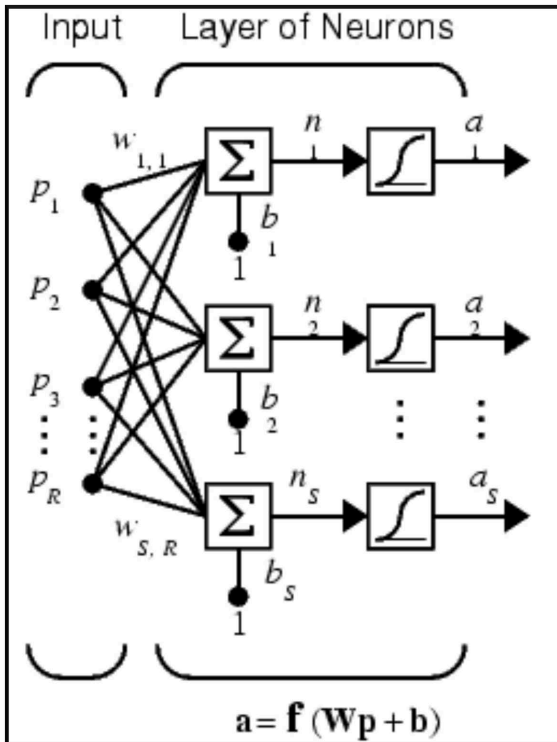
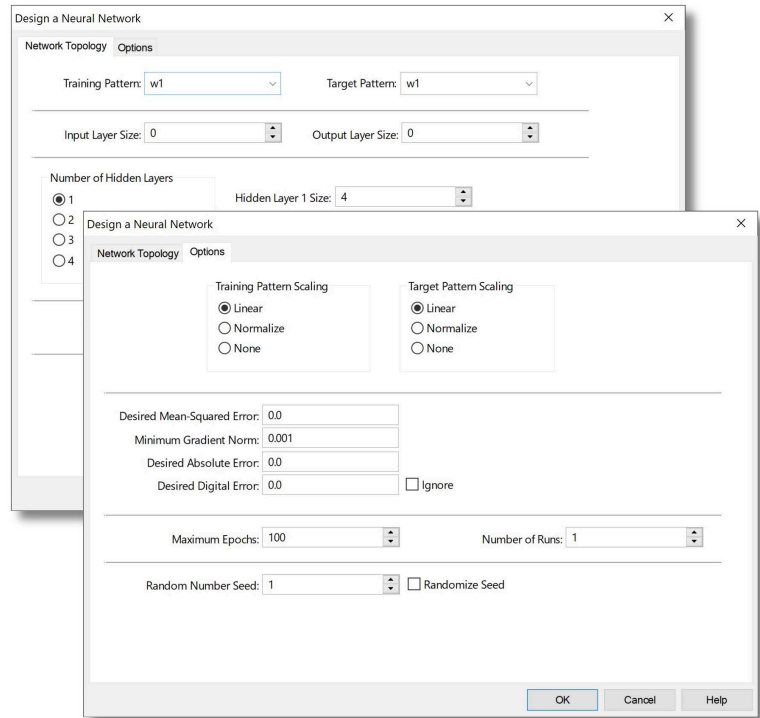
### KEY FEATURES

- Simple User Interface
- Automatic Normalization of Data
- Choice of the Number of Hidden Layers
- Unlimited Input and Output Variables
- Unlimited Number of Runs
- Cross-validation Training to Verify Output Results Simultaneously
- Built-in Protection Against Local Minima Distorting Output Results
- User Selectable Desired Mean Square Error, Minimum Gradient Norm and Desired Absolute Error
- Digital Error, Analog Error, Maximum Error and Gradient Values Post-Training Error Graph Types
- Extract Random Seeds Gives the Values Used to Start a Network and Enables Building another Network with the Same Weights
- Extract Network Weights Returns the Weights and Biases that Define the Network



## Neural Network Module

Neural networks are able to recognize underlying patterns and predict outcomes based on incomplete or inconsistent information. In contrast to expert systems, which use a static set of rules, neural networks are able to learn and adapt to changes in the environment. With DADiSP/NeuralNet, users can build their own artificial neural networks (ANNs) and apply them to achieve more accurate predictions and pattern classifications.



## Neural Network Training

Neural networks resemble the human brain because they can learn. A back-propagation neural network develops its predictive capabilities by being trained on a set of historical inputs and known resulting outputs. The neural net applies random weights to each designated input variable. It then adjusts the weights depending on how closely the actual output values match the desired output values in the training set of historical data. Once the appropriate variable weights have been set that minimize the difference between expected and actual output from the neural net, the neural net can then be applied to new data for classification.

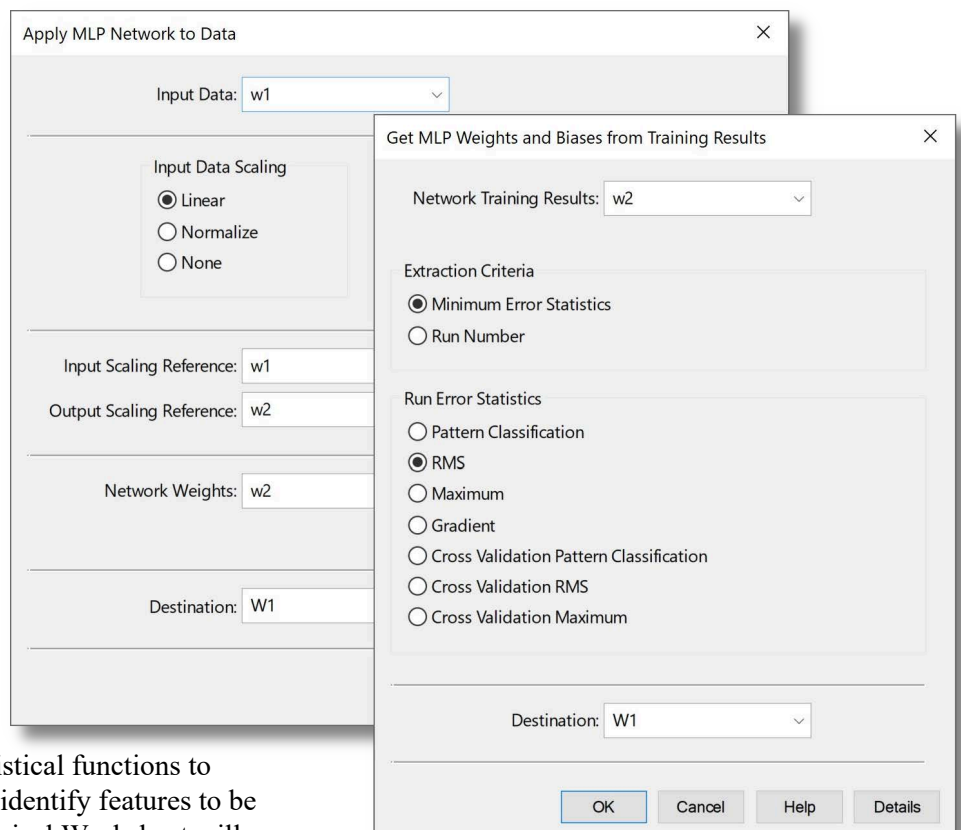
## Back-propagation Learning Algorithm

DADiSP/NeuralNet employs the back-propagation learning algorithm. Back-propagation has become the most widely used neural network paradigm for modeling, forecasting, and classification. To minimize the error in the network, DADiSP/NeuralNet uses a rapid-descent algorithm derived from the Vogl method of locating the global minimum. Since results depend on the initial conditions, the neural net module allows you to train a lot of neural networks on the same data with different initial configurations and pick the best one.

## Powerful Preprocessing Functions

Preprocessing of the data is one of the largest problems in using neural network tools. DADiSP/NeuralNet is fully integrated with DADiSP, so hundreds of analysis functions are available to pre- and post-process neural network data.

DADiSP has mathematical and statistical functions to scale, filter, and process the data to identify features to be learned by the neural network. A typical Worksheet will contain the preprocessing steps, the neural network and the output results. Simply change the input data or initial conditions and each dependent Window is automatically recalculated. You immediately see the effects of your changes on the neural network.



## NeuralNet Functions

DADiSP/NeuralNet includes several functions to create, apply and analyze neural networks.

## NeuralNet Functions

applynet	Apply a neural network to new input data
innorm	Normalize network inputs to +/- 1.0 using mean and variance
inscale	Linearly scale network inputs to +/- 1.0 using min and max
makecvnet	Create a neural network with cross verification
makenet	Create a neural network
nnoptrun	Extract neural network weights based on best error statistics
nnrun	Extract a particular set of post training data for a run
nnseeds	Extract neural network random seed values
nnweight	Extract neural network weights
outnorm	Normalize network output data using a reference series
outscale	Linearly scale network output data using a reference series