Neural Network Toolbox 4

for designing and simulating neural networks

The Neural Network Toolbox extends the MATLAB® computing environment to provide tools for the design, implementation, visualization, and simulation of neural networks. Neural networks are uniquely powerful tools in applications where formal analysis would be difficult or impossible, such as pattern recognition and nonlinear system identification and control. The Neural Network Toolbox provides comprehensive support for many proven network paradigms, as well as a graphical user interface that allows you to design and manage your networks. The toolbox's modular, open, and extensible design simplifies the creation of customized functions and networks.

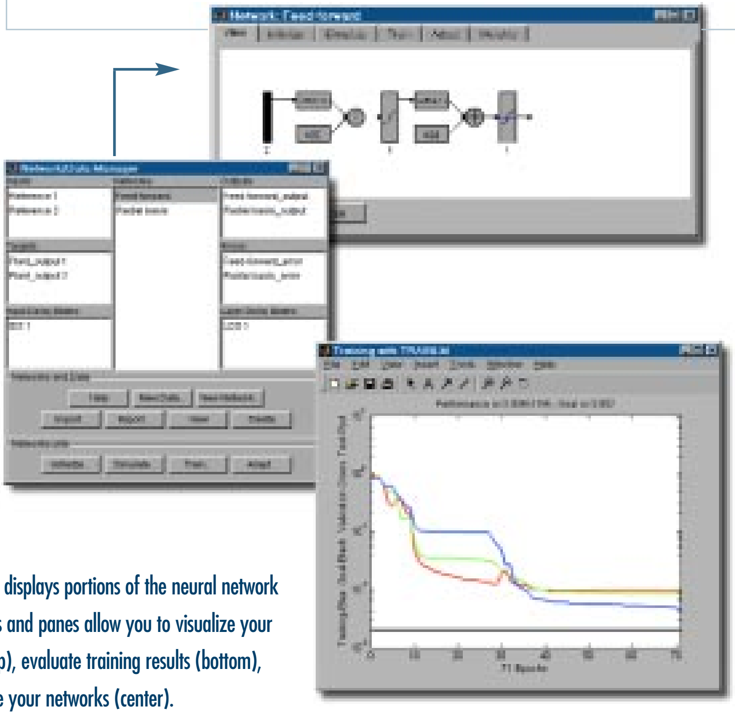
Working with Neural Networks

Inspired by the biological nervous system, neural network technology is being used to solve a wide variety of complex scientific, engineering, and business problems. Commercial applications include investment portfolio trading, data mining, process control, noise suppression, data compression, and speech recognition. Neural networks are ideally suited for such problems because, like their biological counterparts, a neural network can learn, and therefore can be trained to find solutions, recognize patterns, classify data, and forecast events.

Unlike analytical approaches commonly used in fields such as statistics and control theory, neural networks require no explicit model and no limiting assumptions of normality or linearity. The behavior of a neural network is defined by the way its individual computing elements are connected and by the strength of those connections, or weights. The weights are automatically adjusted by training the network according to a specified learning rule until it properly performs the desired task.

KEY FEATURES

- Graphical user interface (GUI) for creating, training, and simulating your neural networks
- Support for the most commonly used supervised and unsupervised network architectures
- A comprehensive set of training and learning functions
- A suite of Simulink® blocks, as well as documentation and demonstrations of control-system applications
- Automatic generation of Simulink models from neural network objects
- Modular network representation, allowing an unlimited number of input sets, layers, and network interconnections
- Pre- and post-processing functions for improving network training and assessing network performance
- Routines for improving generalization
- Visualization functions for viewing network performance



This window displays portions of the neural network GUI. Dialogs and panes allow you to visualize your network (top), evaluate training results (bottom), and manage your networks (center).

Supported Training Functions

`trainb`	Batch training with weight and bias learning rules
`trainbfg`	BFGS quasi-Newton backpropagation
`trainbr`	Bayesian regularization
`trainc`	Cyclical order incremental update
`traincgb`	Powell-Beale conjugate gradient backpropagation
`traincgf`	Fletcher-Powell conjugate gradient backpropagation
`traincgp`	Polak-Ribiere conjugate gradient backpropagation
`traingd`	Gradient descent backpropagation
`traingda`	Gradient descent with adaptive learning rate (lr) backpropagation
`traingdm`	Gradient descent with momentum backpropagation
`traingdx`	Gradient descent with momentum & adaptive lr backpropagation
`trainlm`	Levenberg-Marquardt backpropagation
`trainoss`	One step secant backpropagation
`trainr`	Random order incremental update
`trainrp`	Resilient backpropagation (Rprop)
`trains`	Sequential order incremental update
`trainscg`	Scaled conjugate gradient backpropagation

Because neural networks require intensive matrix computations, MATLAB provides a natural framework for rapidly implementing neural networks and for studying their behavior and application.

Neural Network Toolbox GUI

This tool lets you import potentially large and complex data sets. The GUI also allows you to create, initialize, train, simulate, and manage your networks. Simple graphical representations allow you to visualize and understand network architecture.

Supported Network Architectures

Supervised Networks

Supervised neural networks are trained to produce desired outputs in response to example inputs, making them particularly well suited for modeling and controlling dynamic systems, classifying noisy data, and predicting future events. The Neural Network Toolbox supports the following supervised networks:

- **Feed-forward networks** have one-way connections from input to output layers. They are commonly used for prediction, pattern recognition, and nonlinear function fitting. Supported feed-forward networks include feed-forward backpropagation, cascade-forward backpropagation, feed-forward input-delay backpropagation, linear, and perceptron networks.
- **Radial basis networks** provide an alternative fast method for designing non-linear feed-forward networks. Supported variations include generalized regression and probabilistic neural networks.
- **Recurrent networks** use feedback to recognize both spatial and temporal patterns. Supported recurrent networks include Elman and Hopfield.
- **Learning vector quantization (LVQ)** is a powerful method for classifying patterns that are not linearly separable. LVQ allows you to specify class boundaries and the granularity of classification.

Unsupervised Networks

Unsupervised neural networks are trained by letting the network continually adjust itself to new inputs. They find relationships within data as it is presented and can automatically define classification schemes. The Neural Network Toolbox supports two types of self-organizing unsupervised networks:

- **Competitive layers** recognize and group similar input vectors. By using these groups, the network automatically sorts the inputs into categories.
- **Self-organizing maps** learn to classify input vectors according to similarity. Unlike competitive layers, they also preserve the topology of the input vectors, assigning nearby inputs to nearby categories.

Supported Training and Learning Functions

Training and learning functions are mathematical procedures used to automatically adjust the network's weights and biases. The training function dictates a global algorithm that affects all the weights and biases of a given network. The learning function can be applied to individual weights and biases within a network.

Supported Learning Functions

`learncon`	Conscience bias learning function
`learngd`	Gradient descent weight/bias learning function
`learngdm`	Gradient descent with momentum weight/bias learning function
`learnh`	Hebb weight learning function
`learnhd`	Hebb with decay weight learning rule
`learnis`	Instar weight learning function
`learnk`	Kohonen weight learning function
`learnlv1`	LVQ1 weight learning function
`learnlv2`	LVQ2 weight learning function
`learnos`	Outstar weight learning function
`learnp`	Perceptron weight and bias learning function
`learnpn`	Normalized perceptron weight and bias learning function
`learnsom`	Self-organizing map weight learning function
`learnwh`	Widrow-Hoff weight and bias learning rule

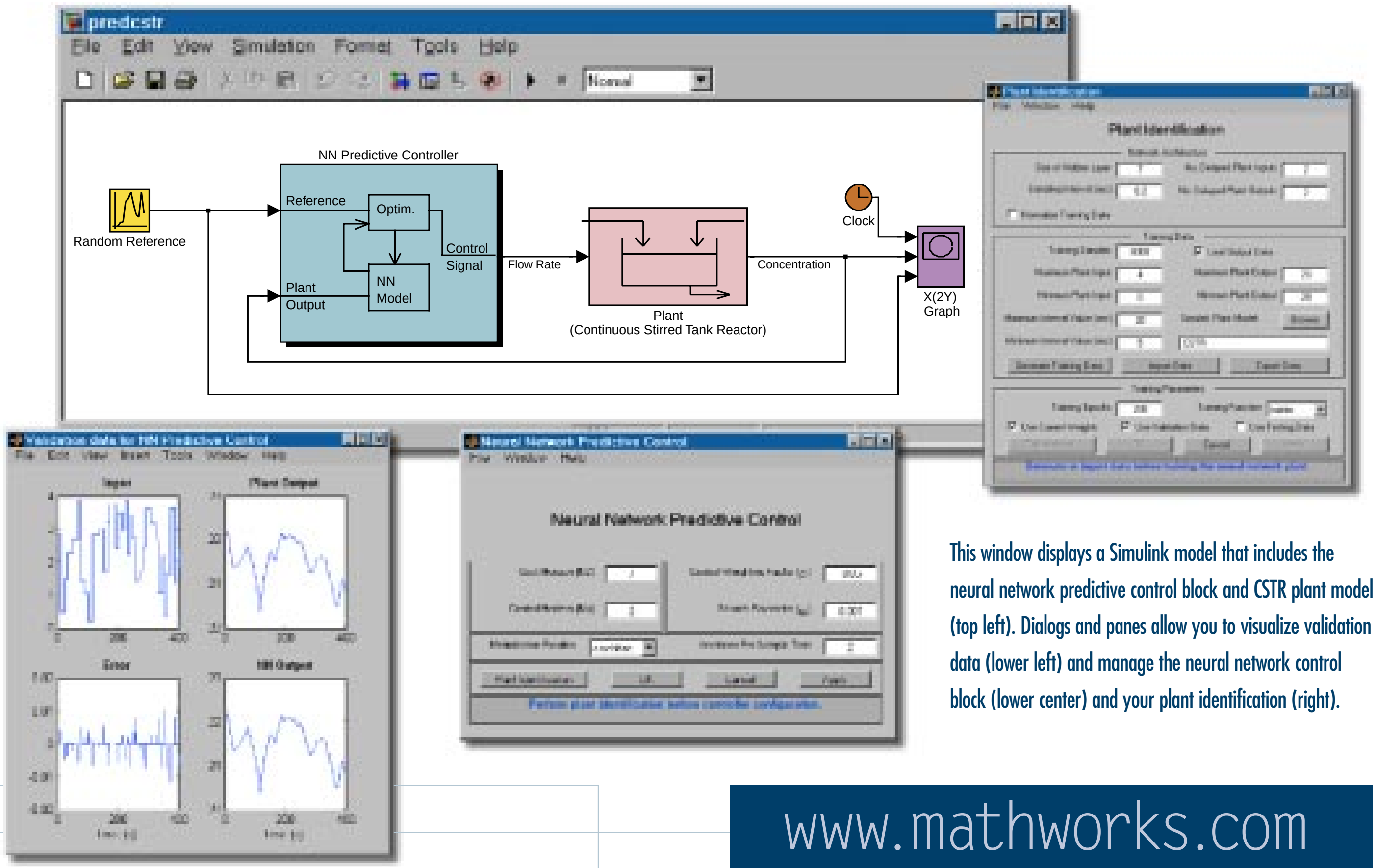
Control System Applications

Neural networks have been successfully applied to the identification and control of nonlinear systems. Included in the toolbox are descriptions, demonstrations, and Simulink blocks for three popular control applications: model predictive control, feedback linearization, and model reference adaptive control.

Model Predictive Control Example

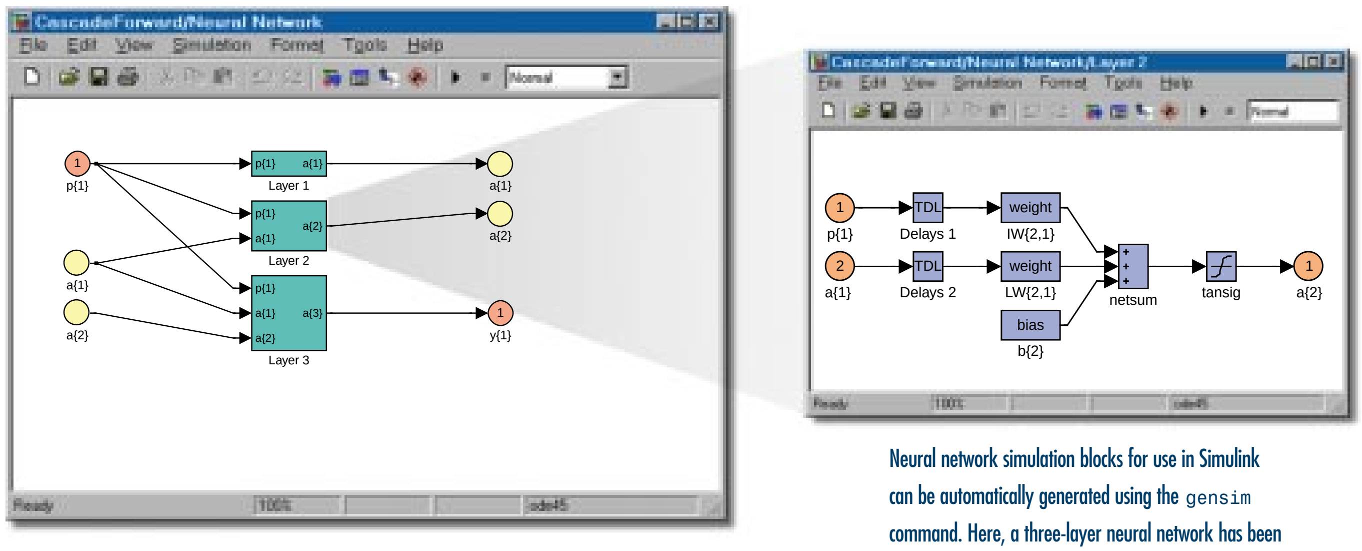
The following example shows the model predictive control of a continuous stirred tank reactor (CSTR). This controller creates a neural network model of a nonlinear plant to predict future plant response to potential control signals. An optimization algorithm then computes the control signals that optimize future plant performance.

You can incorporate neural network control blocks included in the toolbox into your existing Simulink models. By changing the parameters of these blocks you can tailor the network's performance to your application.



This window displays a Simulink model that includes the neural network predictive control block and CSTR plant model (top left). Dialogs and panes allow you to visualize validation data (lower left) and manage the neural network control block (lower center) and your plant identification (right).

www.mathworks.com



Neural network simulation blocks for use in Simulink can be automatically generated using the `gensim` command. Here, a three-layer neural network has been converted into Simulink blocks.

Simulink Support

Once a network has been created and trained, it can be easily incorporated into Simulink models. A simple command (`gensim`) automatically generates network simulation blocks for use with Simulink. This feature also makes it possible for you to view your networks graphically.

Pre- and Post-Processing Functions

Pre-processing the network inputs and targets improves the efficiency of neural network training. Post-processing enables detailed analysis of network performance. The Neural Network Toolbox provides the following pre- and post-processing functions:

- **Principal component analysis** reduces the dimensions of the input vectors.
- **Post-training analysis** performs a regression analysis between the network response and the corresponding targets.
- **Scale minimum and maximum** scales inputs and targets so that they fall in the range [-1,1].
- **Scale mean and standard deviation** normalizes the mean and standard deviation of the training set.

Improving Generalization

Improving the network's ability to generalize helps prevent overfitting, a common problem in neural network design. Overfitting occurs when a network has memorized the training set but has not learned to generalize to new inputs. Overfitting produces a relatively small error on the training set but will produce a much larger error when new data is presented to the network.

The Neural Network Toolbox provides two solutions to improve generalization:

- **Regularization** modifies the network's performance function, the measure of error that the training process minimizes. By changing it to include the size of the weights and biases, training produces a network that not only performs well with the training data, but produces smoother behavior when presented with new data.
- **Early stopping** is a technique that uses two different data sets: the training set, which is used to update the weights and biases, and the validation set, which is used to stop training when the network begins to overfit the data.

Documentation and Examples

The *Neural Network Toolbox User's Guide* was written by Professor Emeritus Howard Demuth and Mark Beale, developers of the Neural Network Toolbox and authors, with Professor Martin Hagen, of *Neural Network Design*. The *User's Guide* is of textbook quality and provides a thorough treatment of neural network architectures, paradigms, and neural network applications. It also includes a tutorial and application examples. Additional demonstrations and application examples are included with the product.

For demos, application examples, tutorials, user stories, and pricing:

- Visit **www.mathworks.com**
- Contact The MathWorks directly

US & Canada	508-647-7000
Benelux	+31 (0)182 53 76 44
France	+33 (0)1 41 14 67 14
Germany	+49 (0)89 995901 0
Spain	+34 93 362 13 00
Switzerland	+41 (0)31 954 20 20
UK	+44 (0)1223 423 200

Visit **www.mathworks.com** to obtain contact information for authorized MathWorks representatives in countries throughout Asia Pacific, Latin America, the Middle East, Africa, and the rest of Europe.

The MathWorks Tel: 508.647.7000 info@mathworks.com www.mathworks.com 8511v04 10/00