Neural Network Toolbox 7.0

Design and simulate neural networks

Neural Network Toolbox™ provides tools for designing, implementing, visualizing, and simulating neural networks. Neural networks are used for applications where formal analysis would be difficult or impossible, such as pattern recognition and nonlinear system identification and control. The toolbox supports feedforward networks, radial basis networks, dynamic networks, self-organizing maps, and other proven network paradigms.

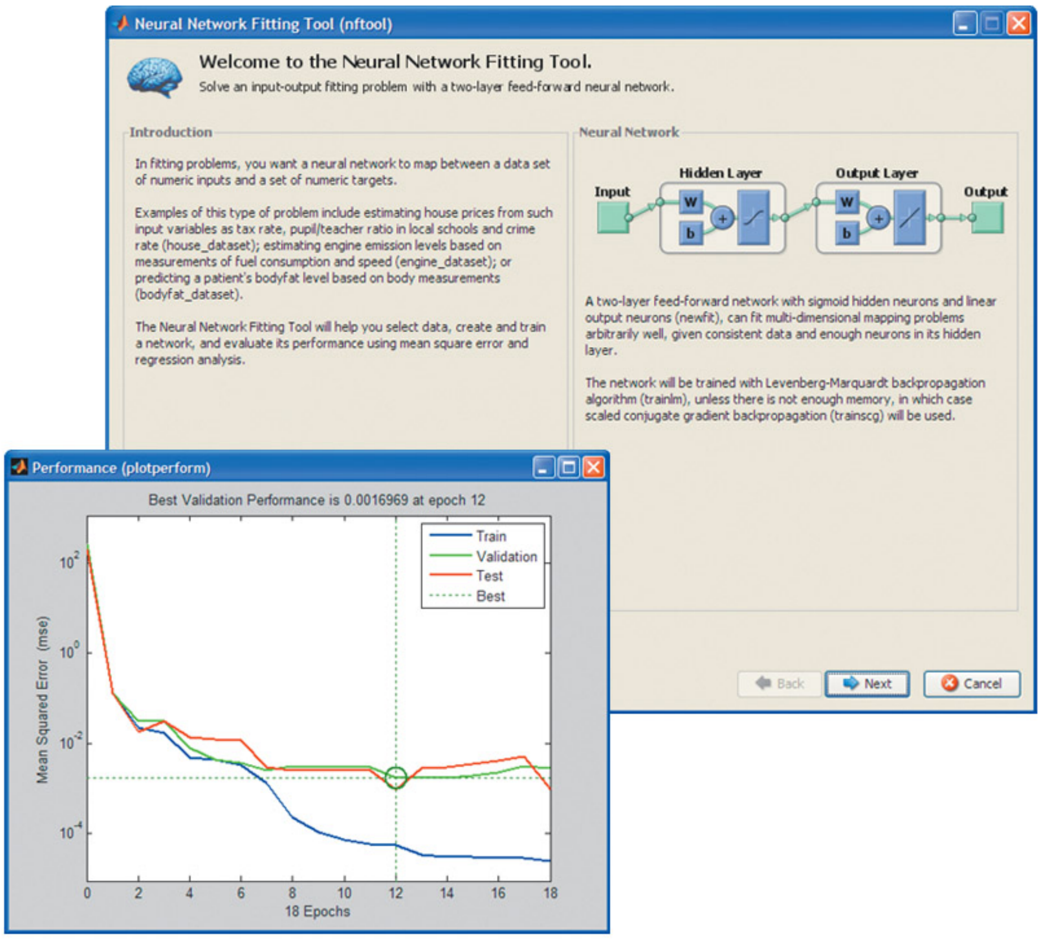
Key Features

- Neural network design, training, and simulation
- Pattern recognition, clustering, and data-fitting tools
- Supervised networks including feedforward, radial basis, LVQ, time delay, nonlinear autoregressive (NARX), and layer-recurrent
- Unsupervised networks including self-organizing maps and competitive layers
- Preprocessing and postprocessing for improving the efficiency of network training and assessing network performance
- Modular network representation for managing and visualizing networks of arbitrary size
- Routines for improving generalization to prevent overfitting
- Simulink® blocks for building and evaluating neural networks, and advanced blocks for control systems applications

Working with Neural Network Toolbox

Like its counterpart in the biological nervous system, a neural network can learn and therefore can be trained to find solutions, recognize patterns, classify data, and forecast future events. 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 performs the desired task correctly.

Neural Network Toolbox includes command-line functions and graphical tools for creating, training, and simulating neural networks. Graphical tools make it easy to develop neural networks for tasks such as data fitting (including time-series data), pattern recognition, and clustering. After creating your networks in these tools, you can automatically generate MATLAB® code to capture your work and automate tasks.


Neural Network Fitting Tool (nftool)

Welcome to the Neural Network Fitting Tool.

Solve an input-output fitting problem with a two-layer feed-forward neural network.

Introduction

In fitting problems, you want a neural network to map between a data set of numeric inputs and a set of numeric targets.

Examples of this type of problem include estimating house prices from such input variables as tax rate, pupil/teacher ratio in local schools and crime rate (house_dataset); estimating engine emission levels based on measurements of fuel consumption and speed (engine_dataset); or predicting a patient's bodyfat level based on body measurements (bodyfat_dataset).

The Neural Network Fitting Tool will help you select data, create and train a network, and evaluate its performance using mean square error and regression analysis.

Neural Network

A two-layer feed-forward network with sigmoid hidden neurons and linear output neurons (newfit), can fit multi-dimensional mapping problems arbitrarily well, given consistent data and enough neurons in its hidden layer.

The network will be trained with Levenberg-Marquardt backpropagation algorithm (trainlm), unless there is not enough memory, in which case scaled conjugate gradient backpropagation (trainscg) will be used.

Performance (plotperform)

Best Validation Performance is 0.0016969 at epoch 12

The Neural Network Fitting Tool (top) and a performance plot (bottom). The Neural Network Fitting Tool guides you through the process of fitting data using neural networks.

Network Architectures

Neural Network Toolbox supports a variety of supervised and unsupervised network architectures. With the toolbox's modular approach to building networks, you can develop custom architectures for your specific problem. You can view the network architecture including all inputs, layers, outputs, and interconnections.

Supervised Networks

Supervised neural networks are trained to produce desired outputs in response to sample inputs, making them particularly well-suited to modeling and controlling dynamic systems, classifying noisy data, and predicting future events.

Neural Network Toolbox supports four types of supervised networks:

- **Feedforward networks** have one-way connections from input to output layers. They are most commonly used for prediction, pattern recognition, and nonlinear function fitting. Supported feedforward networks include feedforward backpropagation, cascade-forward backpropagation, feedforward input-delay backpropagation, linear, and perceptron networks.
- **Radial basis networks** provide an alternative, fast method for designing nonlinear feedforward networks. Supported variations include generalized regression and probabilistic neural networks.

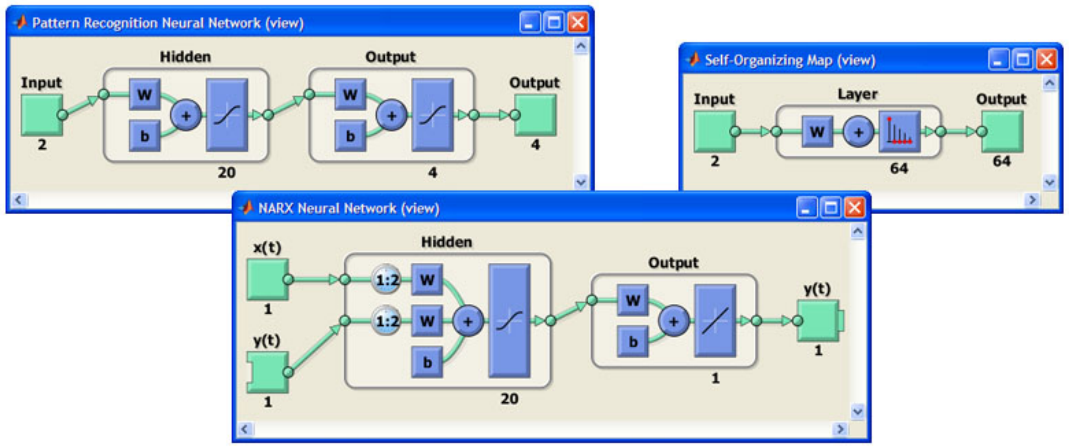
- **Dynamic networks** use memory and recurrent feedback connections to recognize spatial and temporal patterns in data. They are commonly used for time-series prediction, nonlinear dynamic system modeling, and control systems applications. Prebuilt dynamic networks in the toolbox include focused and distributed time-delay, nonlinear autoregressive (NARX), layer-recurrent, Elman, and Hopfield networks. The toolbox also supports dynamic training of custom networks with arbitrary connections.
- **Learning vector quantization (LVQ)** is a powerful method for classifying patterns that are not linearly separable. LVQ lets you 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 and can automatically define classification schemes.

Neural Network Toolbox supports two types of self-organizing, unsupervised networks:

- **Competitive layers** recognize and group similar input vectors, enabling them to automatically sort inputs into categories. Competitive layers are commonly used for classification and pattern recognition.
- **Self-organizing maps** learn to classify input vectors according to similarity. Like competitive layers, they are used for classification and pattern recognition tasks; however, they differ from competitive layers because they are able to preserve the topology of the input vectors, assigning nearby inputs to nearby categories.



Neural networks for pattern recognition (top left), clustering (top right), and time series data fitting (bottom). Neural Network Toolbox lets you design various types of networks and visualize their architectures.

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.

Neural Network Toolbox supports a variety of training algorithms, including several gradient descent methods, conjugate gradient methods, the Levenberg-Marquardt algorithm (LM), and the resilient backpropagation algorithm (Rprop). The toolbox's modular framework lets you quickly develop custom training algorithms that can be integrated with built-in algorithms. While training your neural network, you can use error weights to define the relative importance of desired outputs, which can be prioritized in terms of sample, timestep (for time-series problems), output element, or any combination of these. You can access training algorithms from the command line or via a graphical tool that shows a diagram of the network being trained and provides network performance plots and status information to help you monitor the training process.

A suite of learning functions, including gradient descent, Hebbian learning, LVQ, Widrow-Hoff, and Kohonen, is also provided.

Preprocessing and Postprocessing Functions

Preprocessing the network inputs and targets improves the efficiency of neural network training. Postprocessing enables detailed analysis of network performance. Neural Network Toolbox provides preprocessing and postprocessing functions and Simulink blocks that enable you to:

- Reduce the dimensions of the input vectors using principal component analysis
- Perform regression analysis between the network response and the corresponding targets
- Scale inputs and targets so that they fall in the range [-1,1]
- Normalize the mean and standard deviation of the training set
- Use automated data preprocessing and data division when creating your networks

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 a much larger error when new data is presented to the network.

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 including the sizes of the weights and biases, regularization produces a network that performs well with the training data and exhibits smoother behavior when presented with new data.
- **Early stopping** uses two different data sets: the training set, to update the weights and biases, and the validation set, to stop training when the network begins to overfit the data.

Simulink Support and Control Systems Applications

Simulink Support

Neural Network Toolbox provides a set of blocks for building neural networks in Simulink. All blocks are compatible with Real-Time Workshop®. These blocks are divided into four libraries:

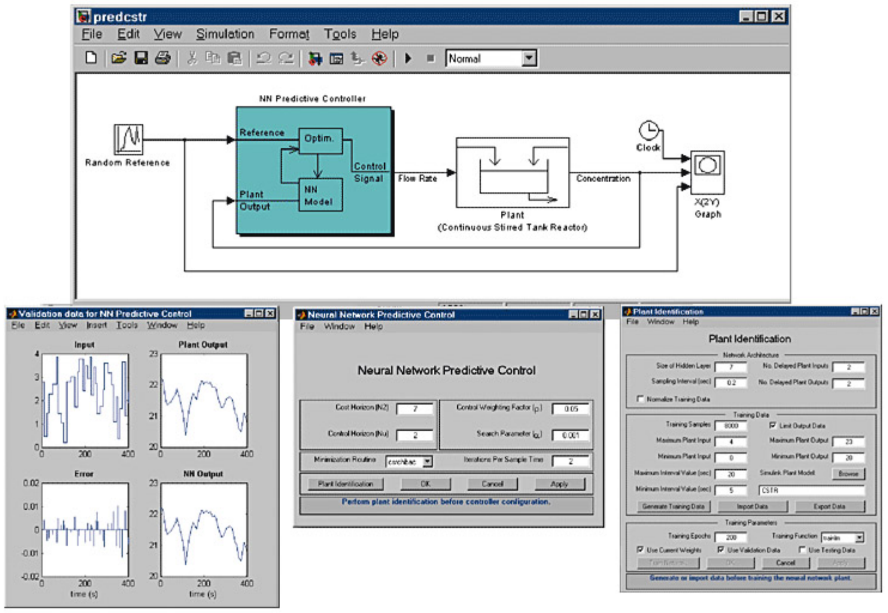
- **Transfer function blocks**, which take a net-input vector and generate a corresponding output vector
- **Net input function blocks**, which take any number of weighted input vectors, weight-layer output vectors, and bias vectors, and return a net-input vector
- **Weight function blocks**, which apply a neuron's weight vector to an input vector (or a layer output vector) to get a weighted input value for a neuron
- **Data preprocessing blocks**, which map input and output data into ranges best suited for the neural network to handle directly

Alternatively, you can create and train your networks in the MATLAB environment and automatically generate network simulation blocks for use with Simulink. This approach also enables you to view your networks graphically.

Control Systems Applications

You can apply neural networks to the identification and control of nonlinear systems. The toolbox includes descriptions, demos, and Simulink blocks for three popular control applications: model predictive control, feedback linearization, and model reference adaptive control.

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



Neural Network Predictive Control

Cost Horizon (N2): 7
Control Weighting Factor (ρ): 0.05
Control Horizon (Nu): 2
Search Parameter (α): 0.001
Minimization Routine: csrchbac
Iterations Per Sample Time: 2

Perform plant identification before controller configuration.

Plant Identification

Network Architecture
Size of Hidden Layer: 7
No. Delayed Plant Inputs: 2
Sampling Interval (sec): 0.2
No. Delayed Plant Outputs: 2
☐ Normalize Training Data

Training Data
Training Samples: 8000
☑ Limit Output Data
Maximum Plant Input: 4
Maximum Plant Output: 23
Minimum Plant Input: 0
Minimum Plant Output: 20
Maximum Interval Value (sec): 20
Simulink Plant Model: CSTR
Minimum Interval Value (sec): 5

Training Parameters
Training Epochs: 200
Training Function: trainlm
☑ Use Current Weights
☑ Use Validation Data
☐ Use Testing Data

Generate or import data before training the neural network plant.

A Simulink model that uses the Neural Network Predictive Controller block with a tank reactor plant model (top). Dialog boxes and panes let you visualize validation data (bottom left) and manage the controller block (bottom center) and your plant identification (bottom right).

Resources

Product Details, Demos, and System Requirements www.mathworks.com/products/neuralnet	**Online User Community** www.mathworks.com/matlabcentral
Trial Software www.mathworks.com/trialrequest	**Training Services** www.mathworks.com/training
Sales www.mathworks.com/contactsales	**Third-Party Products and Services** www.mathworks.com/connections
Technical Support www.mathworks.com/support	**Worldwide Contacts** www.mathworks.com/contact

MathWorks®
Accelerating the pace of engineering and science