
NaN

NaN

The NaN function returns the “Not a Number” value according to the IEEE standards.

Comparison operators do not work with NaN parameters because, by definition, NaN compared to anything, even another NaN, is false. Use **numtype** to test if a value is NaN.

NeuralNetworkRun

NeuralNetworkRun [/Q/Z] **Input=***testWave*, **WeightsWave1=***w1*, **WeightsWave2=***w2*

The NeuralNetworkRun operation uses the interconnection weights generated by NeuralNetworkTrain, and saved in the waves M_Weights1 and M_Weights2, to execute the network for a given input. The input can contain a single run represented by a 1D wave or M runs represented by M columns of a 2D wave. The output of the calculation is saved in the wave W_NNResults or M_NNResults depending on the dimensionality of the input wave. The structure of the network is completely specified by the two weights waves and must match the number of rows in the input wave.

Flags

/Q	Suppresses printing information in the History area.
/Z	No error reporting.

Parameters

Input= <i>testWave</i>	Specifies the input to the neural network. <i>testWave</i> must be a single or double precision wave containing entries in the range [0,1] and have the correct number of rows to match the weights. Execute the network for multiple runs by using a 2D input wave where each column corresponds to a single run. For a 2D input, the result will be stored in M_NNResults with a corresponding column structure.
WeightsWave1= <i>w1</i>	Specifies the interconnection weights between the input and the hidden layer.
WeightsWave2= <i>w2</i>	Specifies the interconnection weights between the hidden layer and the output.

See Also

The NeuralNetworkTrain operation.

NeuralNetworkTrain

NeuralNetworkTrain [/Q/Z] [**keyword = value**]...

The NeuralNetworkTrain operation trains a three-layer neural network. The training produces two 2D waves that store the interconnection weights between the network neurodes. Once you obtain the weights, you can use them with NeuralNetworkRun.

Flags

/Q	Suppresses printing information in the History area.
/Z	No error reporting.

Parameters

keyword is one of the following:

Input= <i>inWave</i>	Specifies the input patterns for training. <i>inWave</i> is a 2D wave where each row corresponds to a single training event and each column corresponds to the input values. The number of rows in <i>inWave</i> (the number of training sets) and in the output wave must be equal. <i>inWave</i> must be single or double precision and all entries must be in the range [0,1].
Iterations= <i>num</i>	Specifies the number of iterations. Default is 10000.
MinError= <i>val</i>	Terminates training when the total error drops below <i>val</i> (default is 1e-8). The total error is normalized, and is defined as the sum of the squared errors divided by the number of training sets times outputs.

Momentum= <i>val</i>	Specifies a coefficient for the back-propagation algorithm. This coefficient adds to the change in a particular weight a contribution proportional to the error in a previous iteration. Default momentum is 0.075.
NHidden= <i>num</i>	Specifies the number of hidden neurodes. You do not need to use the Structure keyword with NHidden because the network is completely specified by the training waves and NHidden.
NReport= <i>num</i>	Specifies over how many iterations (default is 1000) to print the global RMS error to the history area. Ignored with /Q.
Output= <i>outWave</i>	Specifies the expected outputs corresponding to the entries in the input wave. The number of rows in <i>outWave</i> (the number of training sets) and in the input wave must be equal. <i>outWave</i> must be single or double precision and all entries must be in the range [0,1].
LearningRate= <i>val</i>	Sets the network learning rate, which is used in the backpropagation calculation. Default is 0.15.
Restart	Allows specification of your own set of weights as the starting values. Use this to run the training and feed the output weights of one training session as the input for the next.
Structure={ <i>Ni, Nh, No</i> }	Specifies the structure of the network. <i>Ni</i> is the number of neurodes at the input, <i>Nh</i> is the number of hidden neurodes, and <i>No</i> is the number of output neurodes. Structure is unnecessary when using NHidden is because the remaining numbers are determined by the sizes of the input and output waves.
WeightsWave1= <i>w1</i>	Specifies the weights for propagation from the first layer to the second. The 2D wave must be double precision and the dimensions must match the specified neurodes with the same numbers of rows and inputs and with matching numbers of columns and hidden neurodes.
WeightsWave2= <i>w2</i>	Specifies the weights for propagation from the second to the third layer. The 2D wave must be double precision and the dimensions must match the specified neurodes with the same numbers of rows and hidden neurodes and with matching numbers of columns and outputs.

Details

NeuralNetworkTrain is the first half of the implementation of a three-layer neural network in which both in inputs and outputs are taken as normalized quantities in the range [0,1]. Network training is based on back-propagation to iteratively minimize the error between the output and the expected output for any given training set. Training creates in two 2D waves that contain the interconnection weights between the neurodes. M_Weights1 contains the weights between the input layer and the hidden layer and M_Weights2 contains the weights between the hidden layer and the output layer. During the iteration stage, global error information can be printed in the history area.

The algorithm computes the output of the k th neurode by

$$V_k = \left[1 + \exp \left(- \sum_{i=1}^n w_i s_i \right) \right]^{-1},$$

where w_i is the weight corresponding to input i , s_i is the signal corresponding to that input, and n is the number of inputs connected to the neurode.

The total error is defined as the sum (over all training sets and all outputs) of the squared differences between the network outputs and the expected values. The sum is normalized by the product of the number of training sets and the number of outputs. The history reports (see NReport parameter) the square root of the total error (RMS error). The square root of the error computed at the end of the last iteration is stored in the variable V_rms.