FANNTool 1.0
User's Guide

Birol
bluekid70@gmail.com

Michael Schaale
Michael.Schaale@fu-berlin.de

Introduction

An artificial neural network (ANN), usually called "neural network" (NN), is a mathematical model or computational model that tries to simulate the structure and/or functional aspects of biological neural networks. An ANN performs a non-parametric non-linear multivariate multiple regression.

The Fast Artificial Neural Network library (FANN) is a free open source neural network library, which implements multilayer artificial neural networks in C and supports both fully and sparsely connected networks. Cross-platform execution in both fixed and floating point is supported. It includes a framework for easy handling of training data sets. It is easy to use, versatile, well documented, and fast. PHP, C++, .NET, Ada, Python, Delphi, Octave, Ruby, Prolog Pure Data and Mathematica bindings are available.

FANNTool is a GUI to the FANN library which allows its easy usage without the need of programming. This tool enables You to:

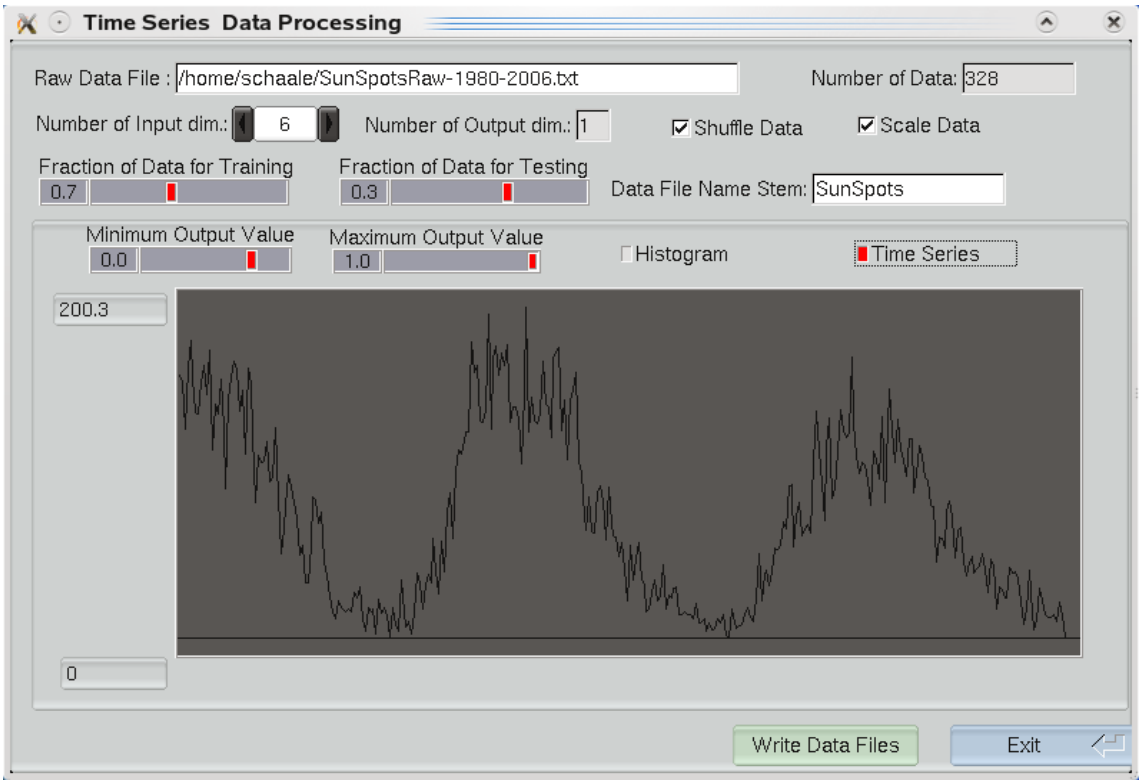
... prepare the data in FANN library standard
... design an ANN
... train the designed ANN
... test the trained ANN
... run the trained ANN

Data example #1 : Time series of sun spots

Let's learn more about *FANNTool* by working through an example: we want to predict the monthly mean of sunspots by using (extrapolating from) past values. The data are given in The ASCII file '*SunSpotsRaw-1980-2006.txt*' lists line-by-line the monthly means of sunspots from January 1980 to October 2006 (= 322 values).

Now launch *FANNTool* and select *Data Processing* to load the data file '*SunSpotsRaw-1980-2006.txt*', the *Time Series Data Processing* dialog opens.

Check the radio button *Time series* and look at a plot mean of sun spots vs. time.



Data Processing reads ASCII raw data and ...

... (optionally) scales them : ANN generally prefer data within a range of (0, 1) or (-1, 1) thus the real-world values usually need to be scaled

... produces a FANN compatible data format.

> The FANN library uses a simple but fixed ASCII data format)
>
> The file must be formatted like
>
> num_train_data num_input_nodes num_output_nodes
> num_input_nodes input data (separated by space) set #1

num_output_nodes output data (separated by space) set #1
.
.
num_input_nodes input data (separated by space) set #num_train_data
num_output_nodes output data (separated by space) set #num_train_data

For example

3 Input 1 Output with 2 data sets:

```
2 3 1
1 1 1
0
1 0 1
1
```

... splits the data into a test and a training data set (option)

... shuffles the data sets (option)

For the data set '*SunSpotsRaw-1980-2006.txt*' we make the following settings:

Number of Input dim. : for example '6' means that 6 monthly mean values are input while the seventh serves as output

Minimum Output Value : desired Minimum Output value after Scaling (here: 0.0)

Maximum Output Value : desired Maximum Output value after Scaling (here: 1.0)

Fraction of data for training : Fraction of the data set that is used for the network training (here: 0.7, i.e. 70 % of the data are allocated for training)

Fraction of data for testing : Fraction of the data set that is used for the network testing (here: 0.3, i.e. 30 % of the data are allocated for training)

Shuffle : shuffle order of the data, i.e. scramble the data sequence

Scale :

For this example

min = 0.0, max = 200.3 (estimated from the data)

(Desired) Minimum Output Value = 0

(Desired) Maximum Output Value = 1

Scaled Value = (Input Value - Minimum) / (Maximum - Minimum)

For this example:

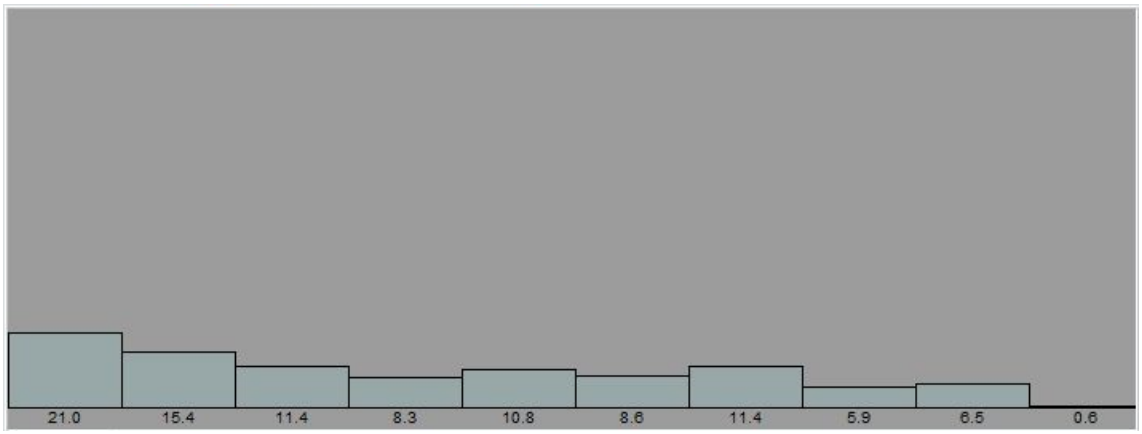
Scaled Value = (Input Value – 0,0) / (200.3 - 0.0)

And the reverse way (descaling):

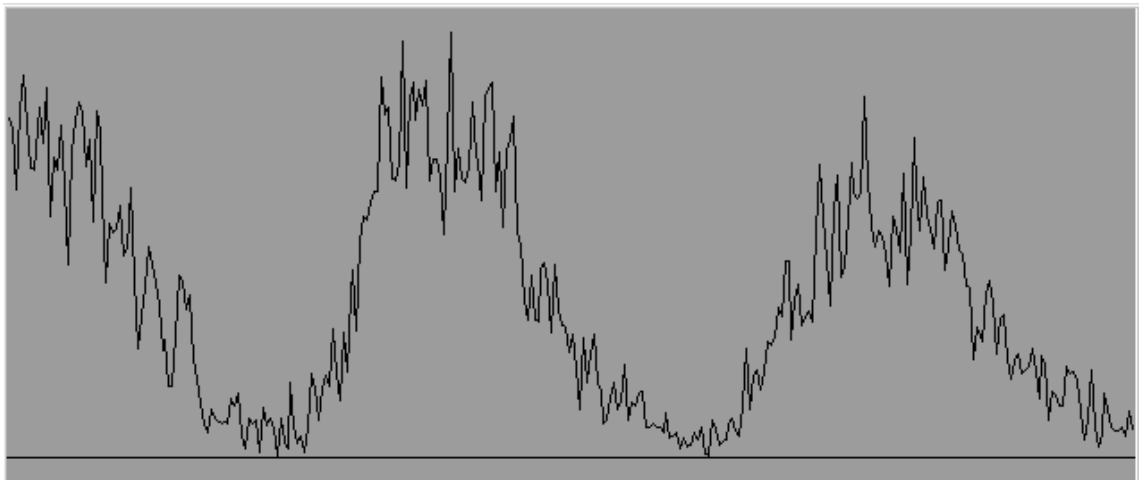
Input Value = (Scaled Value *(200.3 - 0.0)) + 0.0

In the *Time series Data Processing* dialog you can switch between two types of graphs:

(1) Histogram (rather crude sketch)



(2) X-Y plot



Data File Name: Enter path and file name stem for the storage of the FANN formatted training and test data. For example : *'Out'* produces

'Out-train.dat' *: Training Data File*

'Out-test.dat' *: Test Data File*

'Out-scale.txt' *: Text File contain Scaling Parameters*

Data example #2 : multiple regression

There is a another Raw data format which is matrix-like. Each line includes both input and output data. Here the output data are contained in the last columns.

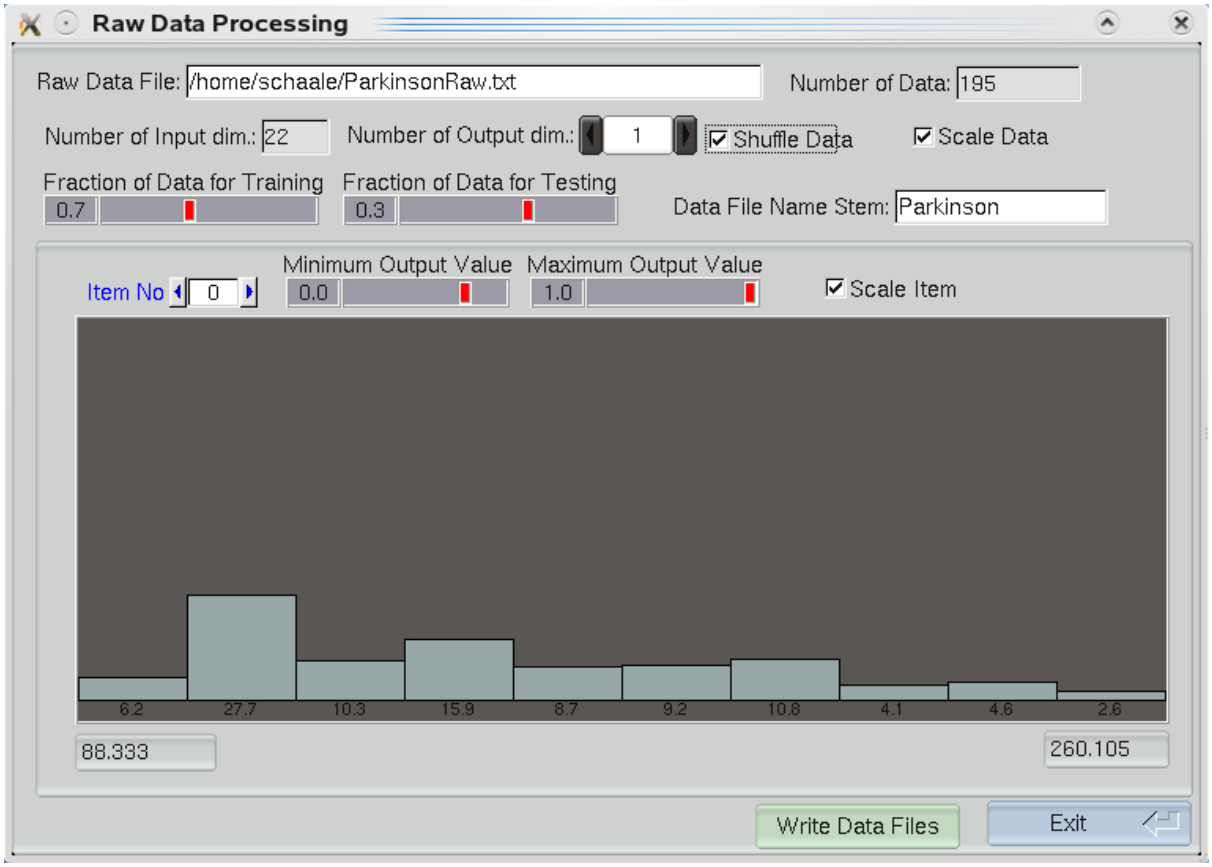
In the example data file '*ParkinsonRaw.txt*' file there are 23 columns, where the first 22 columns are the input dimensions and the 23rd column contains the output data.

The dataset was created by Max Little of the University of Oxford, in collaboration with the National Centre for Voice and Speech, Denver, Colorado, who recorded the speech signals. The original study published the feature extraction methods for general voice disorders.

The file consists of 23 element ASCII records whose fields have the following physical meaning:

Rec	Variable	Meaning
1	MDVP:Fo(Hz)	Average vocal fundamental frequency
2	MDVP:Fhi(Hz)	Maximum vocal fundamental frequency
3	MDVP:Flo(Hz)	Minimum vocal fundamental frequency
4	MDVP:Jitter(%)	\
5	MDVP:Jitter(Abs)	
6	MDVP:RAP	Several measures of variation in fundamental frequency
7	MDVP:PPQ	
8	Jitter:DDP	/
9	MDVP:Shimmer	\
10	MDVP:Shimmer(dB)	
11	Shimmer:APQ3	
12	Shimmer:APQ5	Several measures of variation in amplitude
13	MDVP:APQ	
14	Shimmer:DDA	/
15	NHR	Measure #1 of ratio of noise to tonal components in the voice
16	HNR	Measure #2 of ratio of noise to tonal components in the voice
17	RPDE	Nonlinear dynamical complexity measure #1
18	D2	Nonlinear dynamical complexity measure #2
19	DFA	Signal fractal scaling exponent
20	Spread1	\
21	Spread2	Nonlinear measures of fundamental frequency variation
22	PPE	/
23	Status	Health status of the subject (one) - Parkinson's, (zero) – healthy

Launch *FANNTool,* select *Data Processing* and load the *'ParkinsonRaw.txt'* data file.



In this window (which now looks a bit different from the data example #1) we must specify the number of output dimensions (= # of output nodes).

Number of Output dim. : for this example '1' . It may be greater, i.e. the last n columns are used as output values while the data in the remaining columns serve as input values.

Item No : Each column of the data matrix (= input dimension) can be scaled separately.

The remaining buttons/input fields are the same as in the preceding data example #1.

Setting up a network for the sun spot time series data

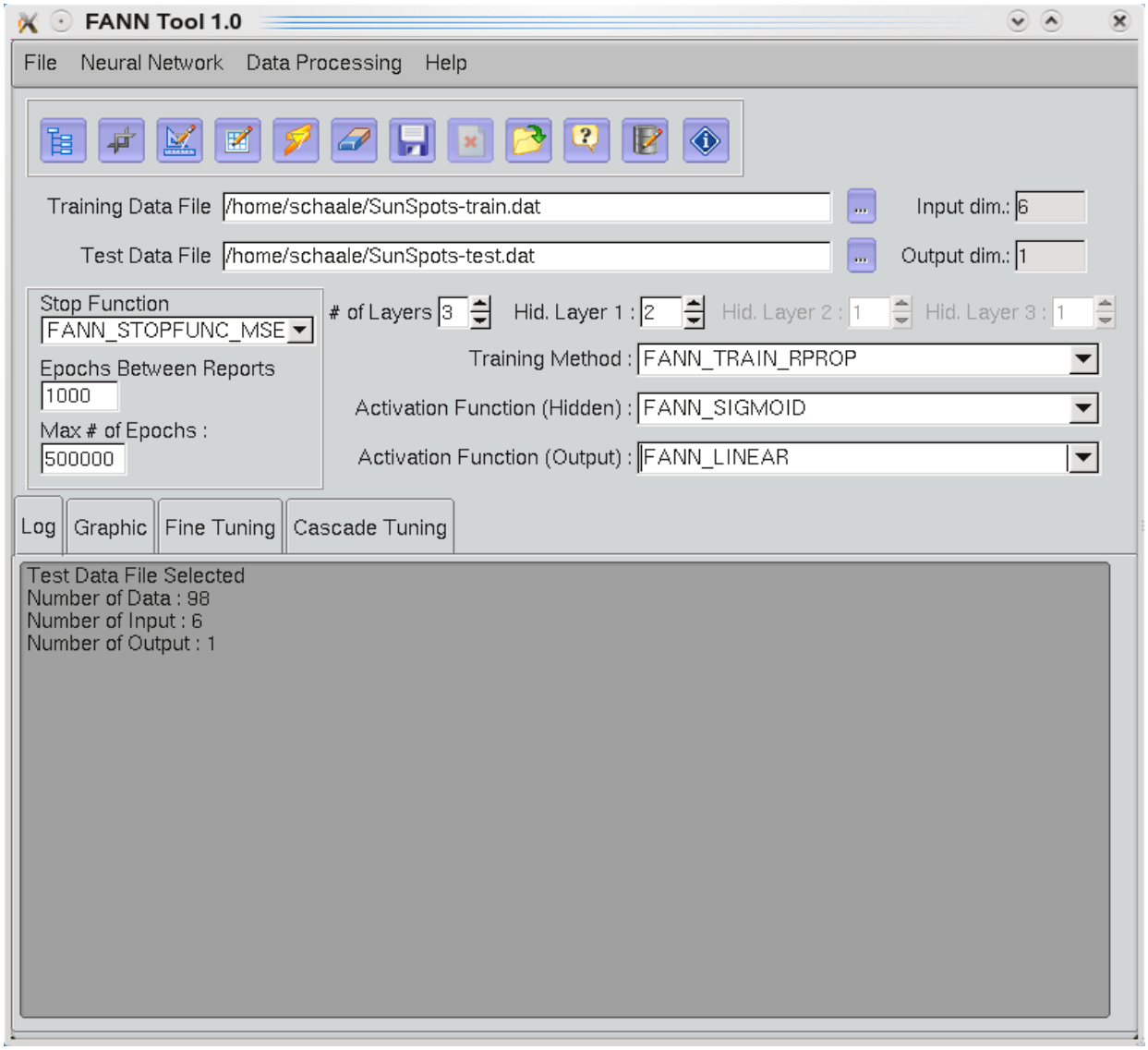
After having converted the raw data into a *FANN* suitable data format one may proceed with the definition of the network.

First load the training and test data :

File → Open Train Data File : Loads training data

File → Open Test Data File : Loads test data

After the reading of the files the number of input nodes (=dimensions) and output nodes is shown (and remains fixed).



Next the hidden layer/s has/have to be specified.

This number can only be altered between 3 and 5, thus the number of hidden layers is constrained to 1, 2 or 3.

# of Layer : Number of layers = Input (1) + Hidden (1 to 3) + Output (1)

so there is a minimum of 3 layers with 1 hidden layer and a maximum of 5 layers with 3 hidden layers. (Rem.: fannlib supports even more hidden layers but there is no need for it).

Hid. layer 1 : # of neurons in the 1st hidden layer

Hid. layer 2 : # of neurons in the 2nd hidden layer (# of Layer ≥ 4)

Hid. layer 3 : # of neurons in the 3rd hidden layer (# of Layer = 5)

Training Method : You can choose one of the following training algorithms:

FANN_TRAIN_INCREMENTAL: Standard backpropagation algorithm in pattern mode: the weights are updated after each training pattern. This means that the weights are updated many times during a single epoch. For this reason some problems, will train very fast with this algorithm, while other more advanced problems will not train very well.

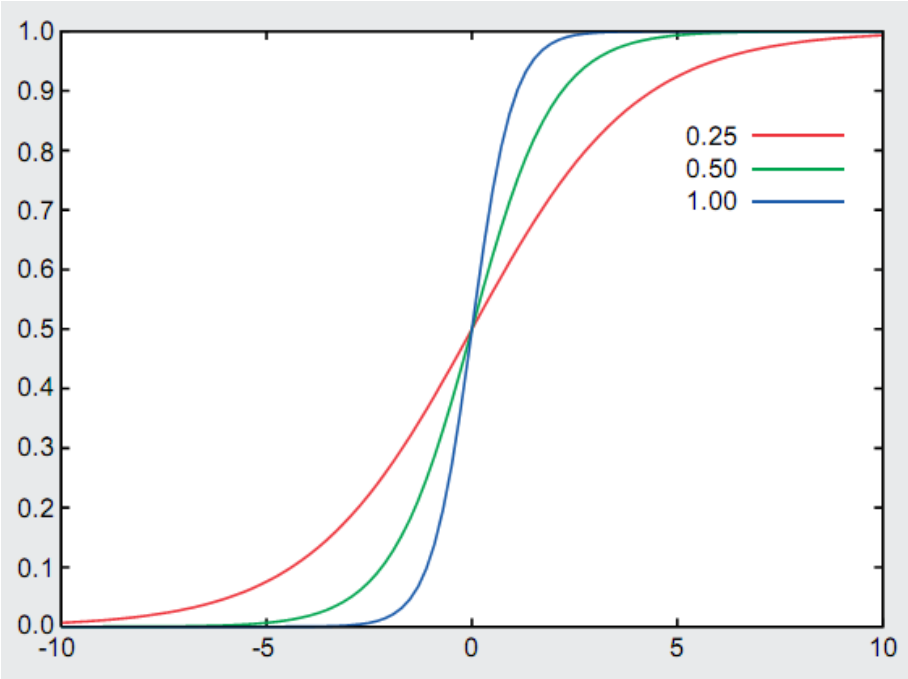
FANN_TRAIN_BATCH: Standard backpropagation algorithm in batch mode: the weights are updated after calculating the mean square error for the whole training set. This means that the weights are only updated once during an epoch. This raises the problem the algorithm will converge slower. But since the mean square error is calculated more correctly than in incremental training, some problems will reach a better solutions with this algorithm.

FANN_TRAIN_RPROP: A more advanced batch training algorithm which achieves good results for most problems. The RPROP training algorithm is adaptive, and does therefore not use a fixed learning_rate. Some other parameters can however be set to change the way the RPROP algorithm works, but it is only recommended for users with insight in how the RPROP training algorithm works. The RPROP training algorithm is described by [Riedmiller and Braun, 1993], but the actual learning algorithm used here is the iRPROP- training algorithm which is described by [Igel and Husken, 2000] which is a varient of the standard RPROP training algorithm.

FANN_TRAIN_QUICKPROP: A more advanced batch training algorithm which achieves good results for many problems. The quick propagation (quickprop) training algorithm uses the learning_rate parameter along with other more advanced parameters, but it is recommended to change these advanced parameters by users with insight in how the quickprop training algorithm works only. The quickprop training algorithm is described by [Fahlman, 1988].

Activation Function Hidden : Hidden Layer Activation Function
Activation Function Output : Output Layer Activation Function

The graphic below shows a sketch of the sigmoid function with different (0.25, 0.5, 1) steepness values.



In the *Fine Tuning* section you will see the *Steepness* option. The figure above illustrates the effect of varying this value on the steepness of the transition from the function's (here: FANN_SIGMOID) minimum to its maximum value. The smaller the steepness becomes the more linear the transfer function behaves. And reversely the larger the steepness is chosen the more the smooth sigmoid function approaches a step function. For more details see here. *FANNTool* allows to choose different step functions from a huge variety (see FANN documentation http://leenissen.dk/fann/html/files/fann_data-h.html#fann_activationfunc_enum).

Network training

Neural Network → Detect → Optimum Training Algorithm: Each possible Training Algorithm is used for several epochs. All other parameters are fixed and the weight initialization is identical. The Training Algorithm showing the lowest MSE is picked.

Neural Network → Detect → Optimum Activation Functions: Each possible Activation Function is used for several epochs. All other parameters are fixed and the weight initialization is identical. The Activation Function yielding the lowest MSE is picked.

Neural Network → Train → Normal: Fixed topology training. The size and topology of the ANN is determined in advance and the training alters the weights in order to minimize the difference between the desired output values and the actual output values. This kind of training is supported by fann_train_on_data.

Neural Network → Train → Cascade: Evolving topology training. The training starts with an empty ANN, only consisting of input and output neurons. Hidden neurons and connections are added during training, in order to reach the same goal as for fixed topology training. This kind of training is supported by FANN Cascade Training.

Stop Function : Stop criterion used during training.

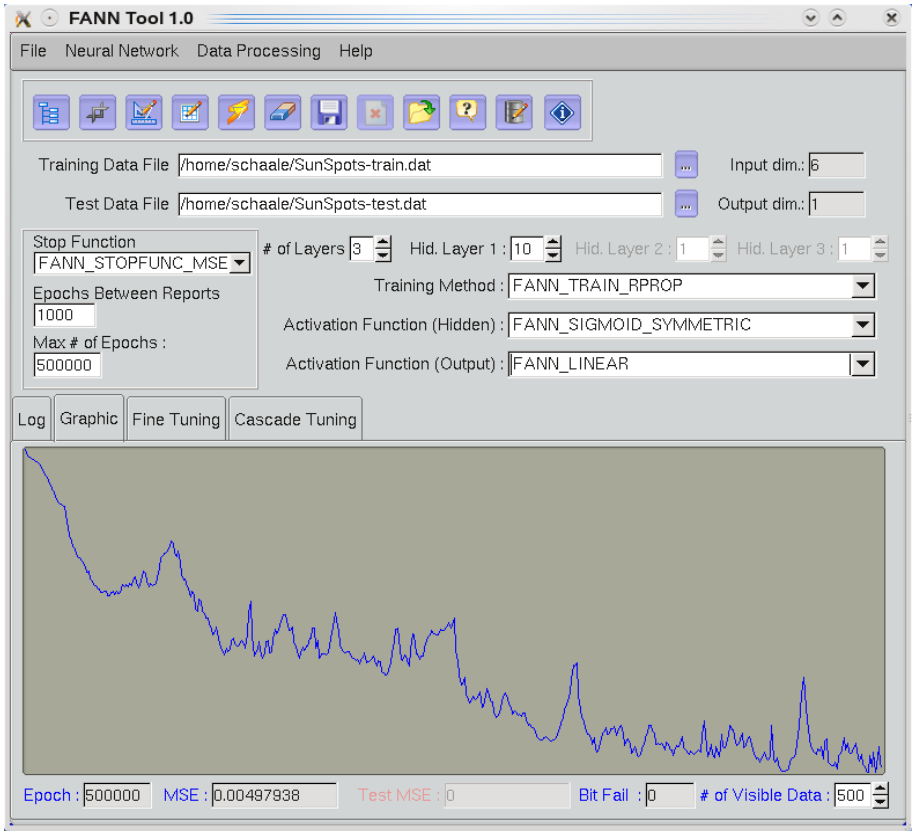
FANN_STOPFUNC_MSE: Stop criterion is Mean Square Error (MSE) value.

FANN_STOPFUNC_BIT: Stop criterion is the number of bits that fail. 'Number of bits' means the number of output neurons which differ more than the bit fail limit.

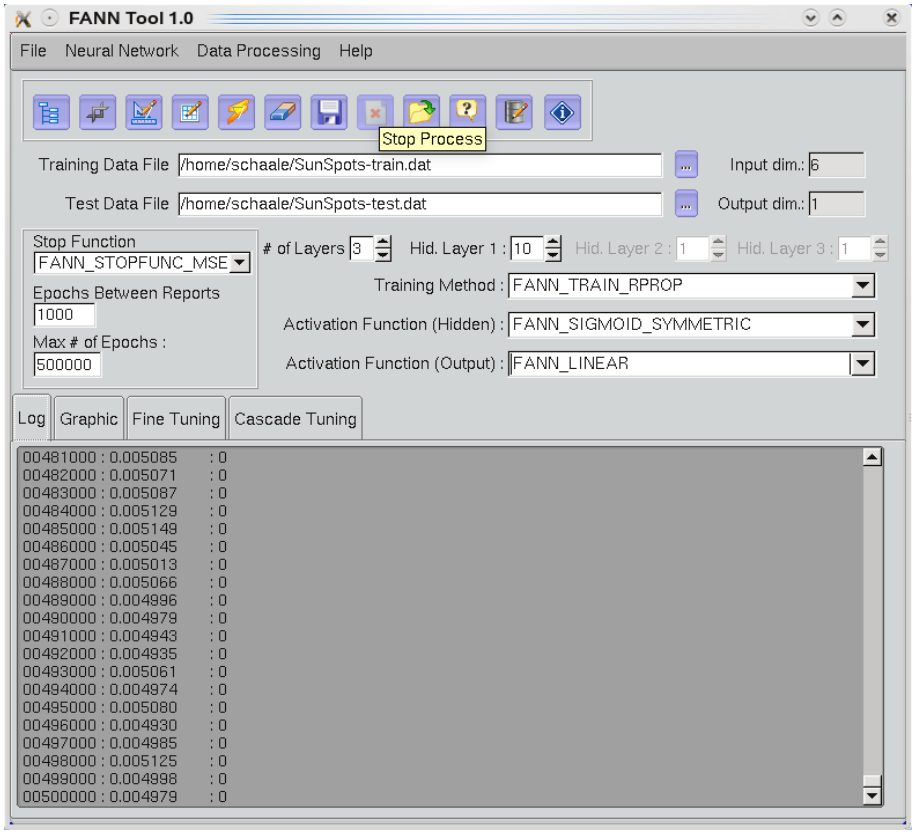
Epoch Between Reports : The number of epochs between printing the status to the Log (file) and Graphic (window)

Max Epoch : The maximum number of training epochs

The Main Window after the training with the Graph tab ....



.... and with the Log tab opened.



Tabs in the main window

Log (see above)

All the text output written this section.

Clear Log clears the log screen

Save Log saves the content of the Log window to a text file.

Open Log optionally loads the log from a (saved) text file

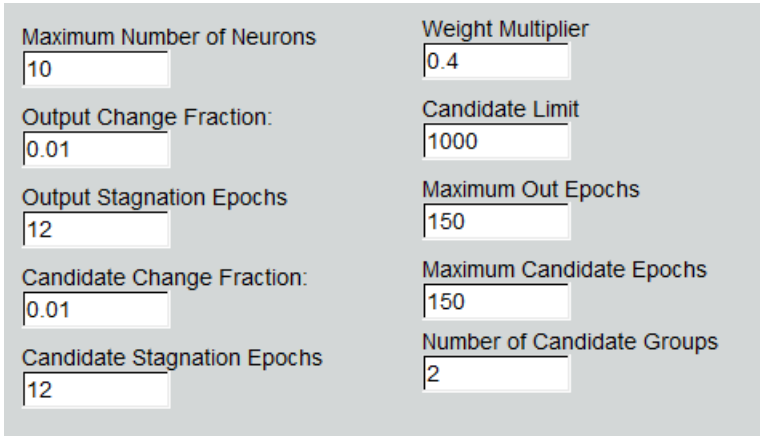
Graphic (see above)

Plots a MSE vs. Epoch graph during the training. If OCS is not activated the plotted MSE is the MSE for the training data only. İf OCS is activated the mean of the MSE for the training data and the MSE for testing is used. You can also see the numeric MSE values at the bottom of the graphic window.

Cascade Tuning

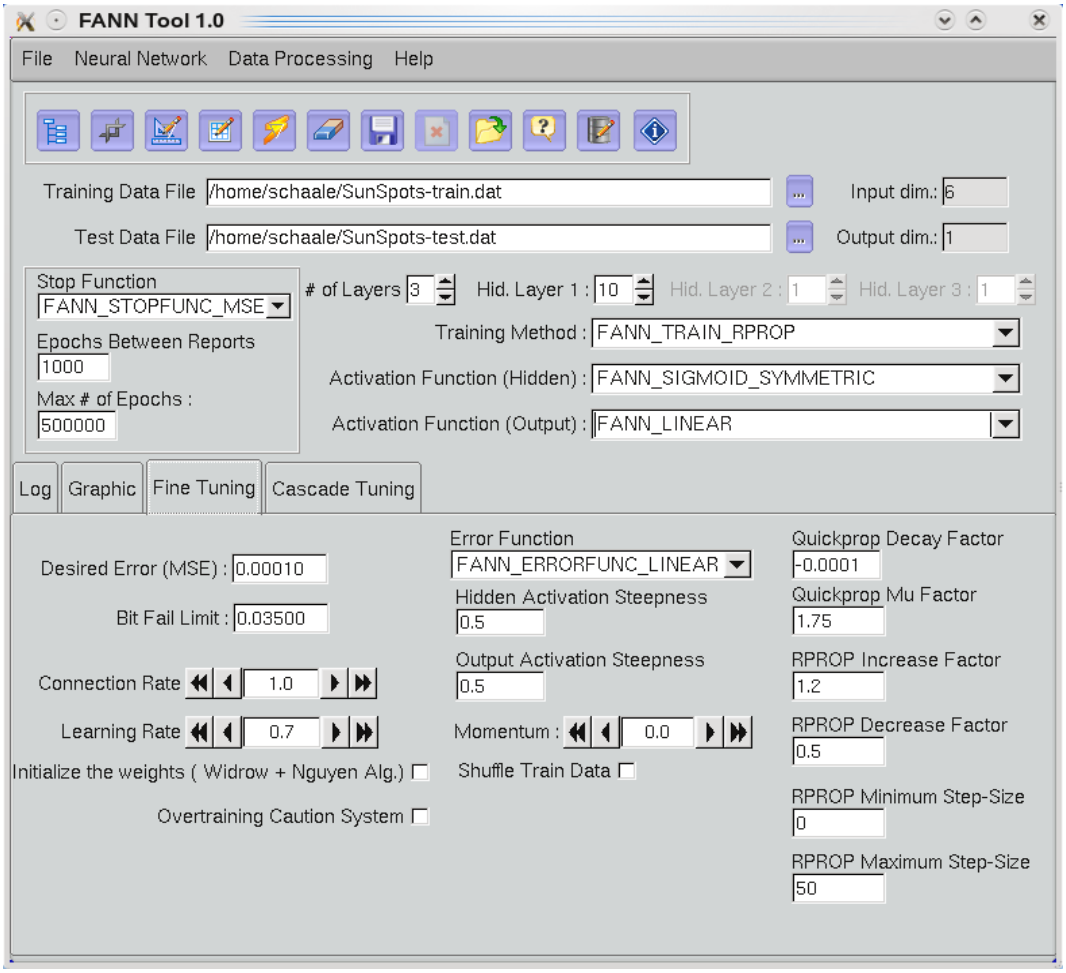
Cascade training differs from ordinary training in the sense that it starts with an empty neural network and then adds neurons one by one, while it trains the neural network. The main benefit of this approach, is that you do not have to guess the number of hidden layers and neurons prior to training, but cascade training have also proved better at solving some problems.

The basic idea of cascade training is that a number of candidate neurons are trained separate from the real network, then the most promising of these candidate neurons is inserted into the neural network. Then the output connections are trained and new candidate neurons is prepared. The candidate neurons are created as shortcut connected neurons in a new hidden layer, which means that the final neural network will consist of a number of hidden layers with one shortcut connected neuron in each. For detailed information about the parameters consult the FANN documentation (http://leenissen.dk/fann/html/files/fann_cascade-h.html#Parameters).



FineTuning

In this tab several parameters which are linked to the learning stage can be (if necessary) adapted and fine-tuned.



The meaning of the parameters is briefly summarized here:

Quickprop Decay Factor: The decay is a small negative valued number which is the factor that the weights should become smaller in each iteration during quick propagation training. This is used to make sure that the weights do not become too high during training.

The default decay is -0.0001.

Quickprop Mu Factor: The mu factor is used to increase and decrease the step-size during quickprop training. The mu factor should always be above 1, since it would otherwise decrease the step-size when it was suppose to increase it.

The default mu factor is 1.75.

RPROP Increase Factor: The increase factor is a value larger than 1, which is used to increase the step-size during RPROP training.

The default increase factor is 1.2.

RPROP Decrease Factor: The decrease factor is a value smaller than 1, which is used to decrease the step-size during RPROP training.

The default decrease factor is 0.5.

RPROP Minimum Step-Size: The minimum step-size is a small positive number determining how small the minimum step-size may be.

The default value delta min is 0.0.

RPROP Maximum Step-Size: The maximum step-size is a positive number determining how large the maximum step-size may be.

The default delta max is 50.0.

Desired Error (MSE): desired mean squared error of the network.

Bit Fail Limit: The number of fail bits, i.e. the number of output neurons which differ more than the bit fail limit.

Hidden Activation Steepness:
Output Activation Steepness:

The steepness of an activation function describes how fast the activation function goes from the minimum to the maximum. A large value yields a steep activation function.

When training neural networks where the output values should be at the extremes (usually 0 and 1, depending on the activation function), a steep activation function can be used (e.g. 1.0).

The default activation steepness is 0.5.

Learning Rate: The learning rate is used to determine how aggressive training should be for some of the training algorithms (FANN_TRAIN_INCREMENTAL, FANN_TRAIN_BATCH, FANN_TRAIN_QUICKPROP). Do however note that it is not used in FANN_TRAIN_RPROP.

The default learning rate is 0.7

Momentum: The learning momentum can be used to speed up FANN_TRAIN_INCREMENTAL training. A too high momentum will however not benefit training. Setting momentum to 0 will be the same as not using the momentum parameter. The recommended value of this parameter is between 0.0 and 1.0.

The default momentum is 0.

Connection Rate: The connection rate controls how many connections there will be in the network. If the connection rate is set to 1, the network will be fully connected, but if it is set to 0.5 only half of the connections will be set.

Shuffle Train Data: Shuffles training data, randomizing the order. This is recommended for incremental training, while it have no influence during batch training.

Initialize the weights (Widrow+Nguyen Alg.): Initialize the weights using the Widrow + Nguyen's algorithm. This function behaves similar to *fann_randomize_weights()*. It will use the algorithm developed by Derrick Nguyen and Bernard Widrow to set the weights in such a way as to speed up training. This technique is not always successful, and in some cases it can be less efficient than a purely random initialization.

Error Function: Error function to be minimized during training.

FANN_ERRORFUNC_LINEAR: Standard linear error function.

FANN_ERRORFUNC_TANH: Tanh error function, usually better but can require a lower learning rate. This error function aggressively targets outputs that differ much from the desired value, while not targeting outputs that differ slightly only. This error function is not recommended for cascade training and incremental training.

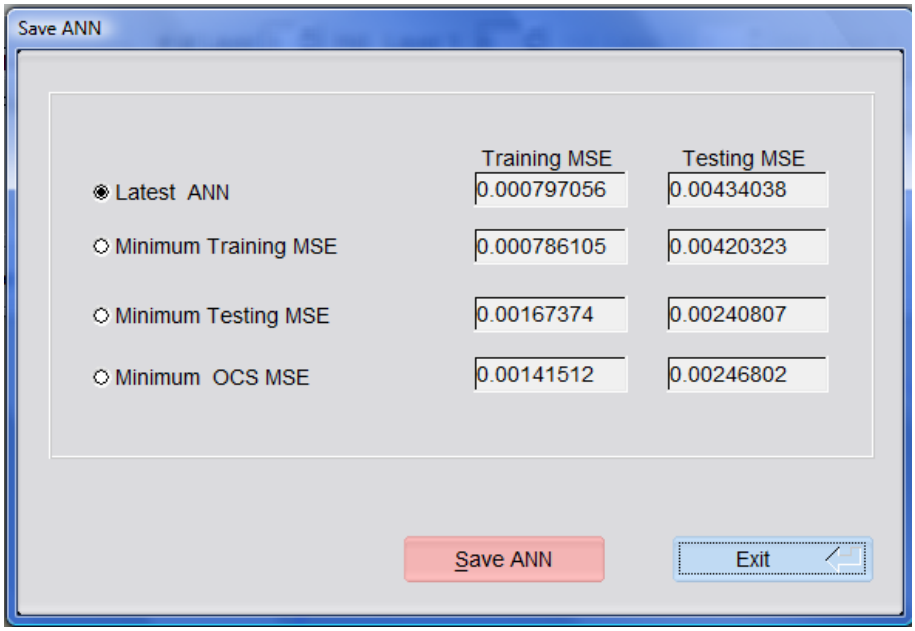
Overtraining Caution System (OCS): When selected a test data (=data being independent of the ANN training) file has to be loaded too. During the training you can watch the test data's and the training data's MSE values. The training will stop if the overall MSE (=MSE OCS), i.e. the arithmetic mean of the test and training data MSE,

$$\text{MSE OCS} = (\text{MSE(Training)} + \text{MSE (Test)}) / 2$$

will show an increase. An increasing MSE OCS indicates a decreasing generalization capability of the network. During the training the graph of the MSE OCS is plotted in the *Graphic* window.

Saving trained Neural Networks

After the regular end of a training or when it was stopped  by the user, the program asks the user wether to save the network:



When answering with 'yes' the *Save ANN* dialog opens. Simply choose the ANN You want to save then click *Save ANN:*

Latest ANN: ANN when training stopped

Minimum Training MSE: minimum MSE value with Training Data During training process

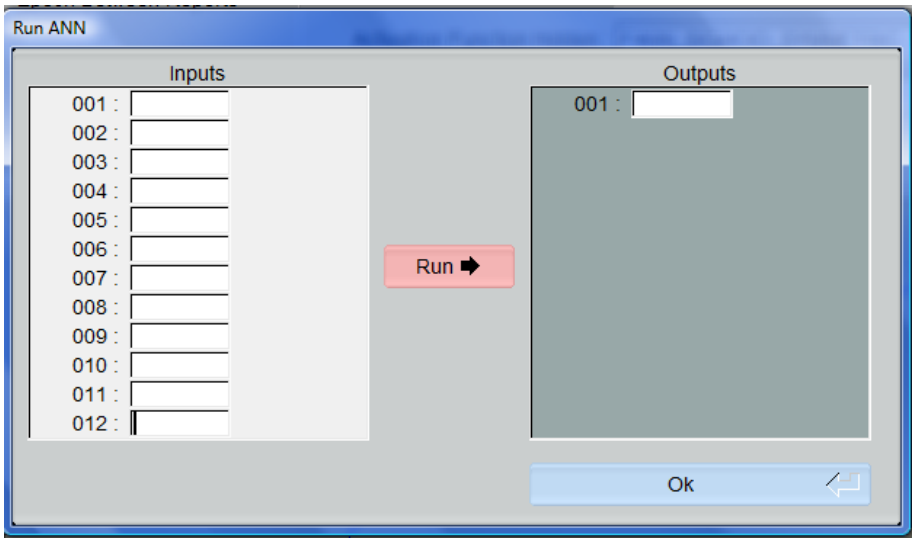
Minimum Testing MSE: minimum MSE value with Testing Data During training process (if OCS was activated)

Minimum OCS MSE: minimum mean MSE value with Testing and Training Data During training process (if OCS was activated)

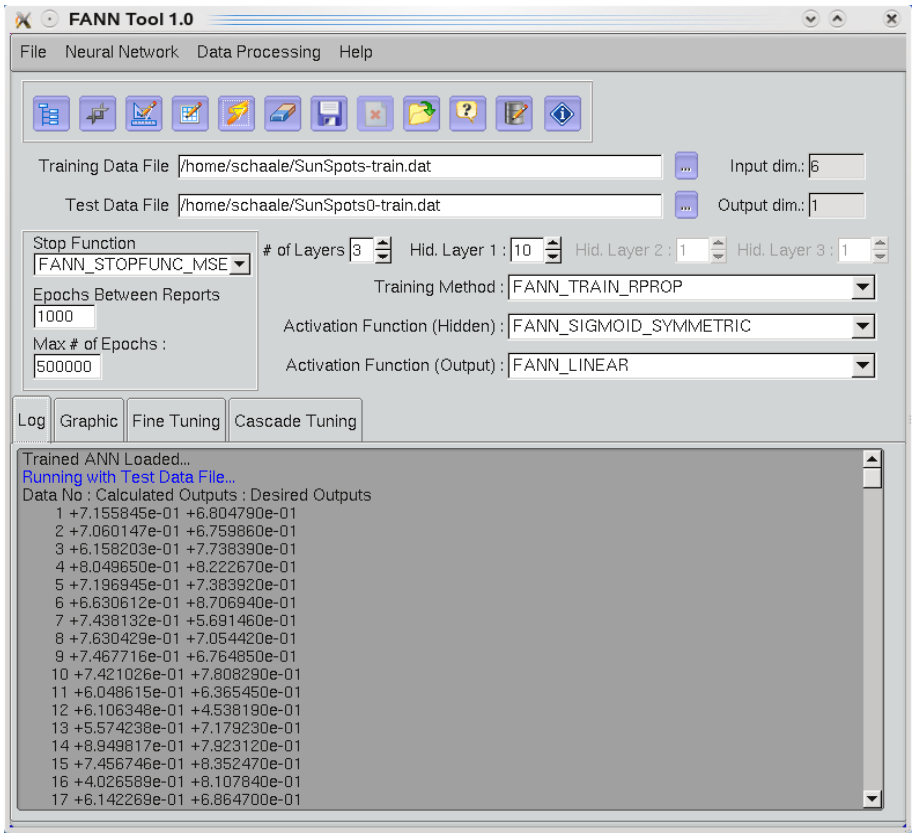
A trained network can be *Run*  inside *FANNTool* in two ways

Run : One can *Run* a trained ANN within the *FANNTool* by using 2 methods:

First method: **Neural Network** → **Run** → **Normal** : Simply select saved ANN file. Then the *Run ANN* dialog opens. Add input values and then click the *Run* button and see the answer of the ANN in the output section. All inputs must be scaled. Output received there are descaled to real values.

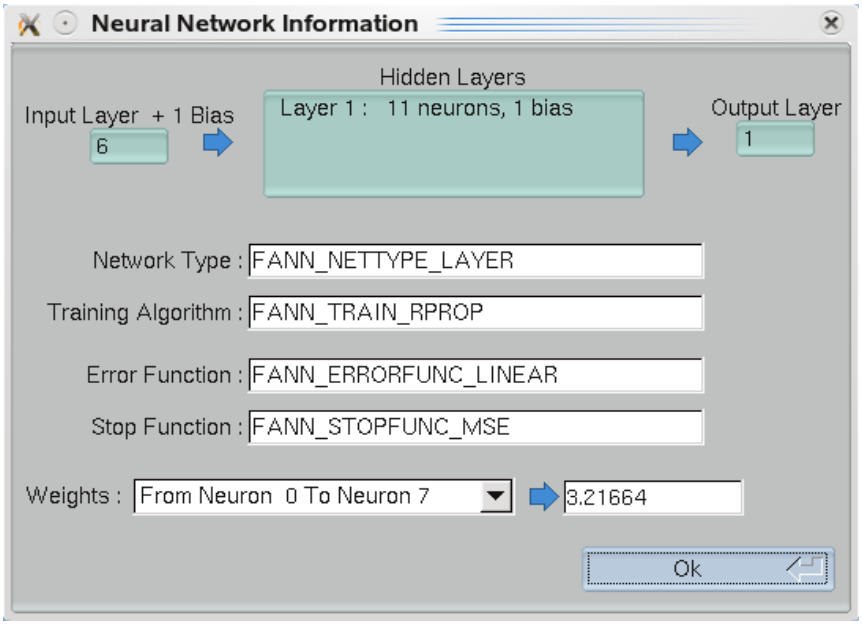


Second method: Write scaled data in a FANN compatible data format file (see above). Now load this file to the test data section(**File** → **Open Test Data File**). Now use the stored ANN: **Neural Network** → **Run** → **with File**. The result of the recall is written to the *Log* screen.



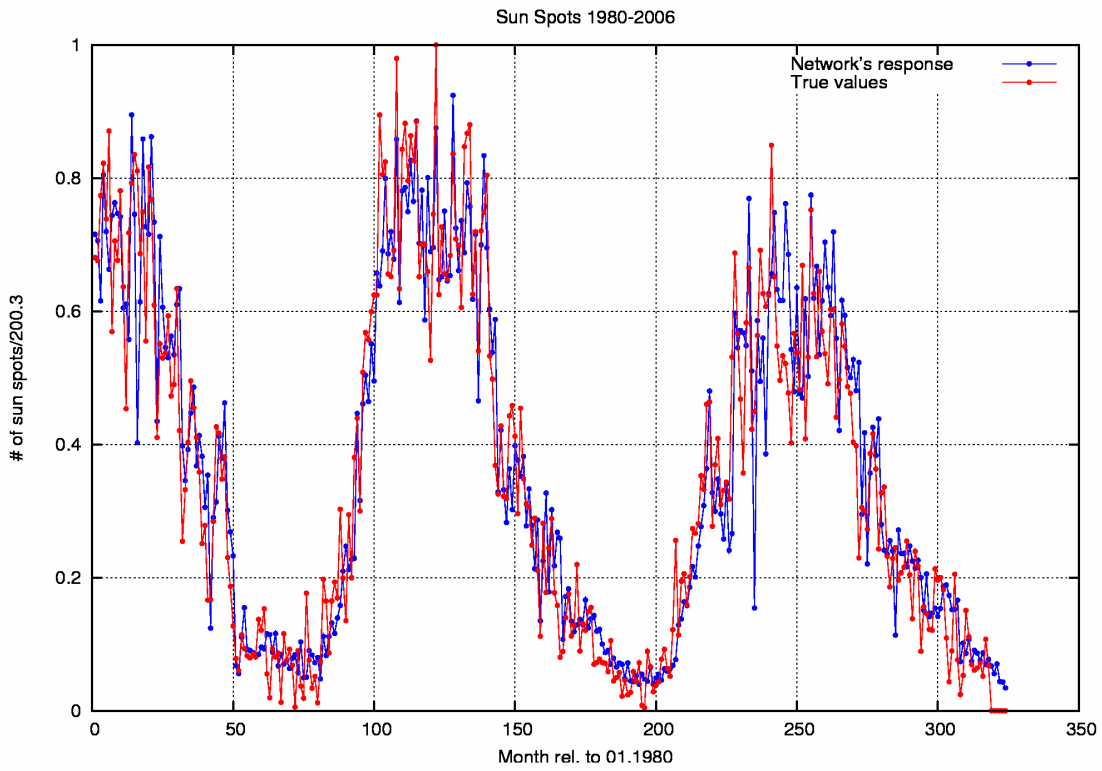
Neural Network Information

Obtain information  about the saved ANN files: simply select an ANN file and the *Neural Network information* dialog opens up and You can see the network's structure, some parameters and selected weight of the ANN .



Recall result for the sun spot time series example

After the training of the sun spot data the network was saved to the file *'SunSpots.net'*. Next the whole(!) raw data was imported again and stored in a scaled version without(!) shuffling the data. The *Fraction of Data for Training* field was set to 1.0 prior to export. Finally the *'*-train.dat'* data set was loaded into the *Test Data File* field. Next **Neural Network → Run → with File** was selected and the trained network *'SunSpot.net'* was selected. The network's responses together with the desired responses appeared in the Log window and were stored in an ASCII data file. Those data maybe imported into *MS Excel* or *OpenOffice Calc* for a rescaling or they may be plotted directly with *gnuplot* (see below).



Concluding remarks

The range of possible applications of Artificial Neural Networks is huge and nowadays they are used quite frequently. The software *FANNTool* will hopefully encourage more scientists and non-scientists to look into the subject. This guide briefly describes how to use this program and it does not claim to be complete or absolutely clear nor will the software be bug-free. If You encounter any problems or if You have any suggestions please feel free to contact us by email, we will be happy if we can help You.

All data and net files mentioned above are distributed with this documentation. We recommend to use these examples to learn more about *FANNTool* and to practice it intensively.

More examples can be pulled from my Blog which is written in turkish(!) language

Letter recognition
Nodule count determination in spherical cast irons
Classification of heart disease
Eye finding by using an ANN
Blood donation prediction by using ANN
ANN playing Tic Tac Toe
FANNTool publications

BlueKid