WinNN version 0.93

WinNN - Windows Neural Networks, a neural network package that will train a fully connected feed-forward   networks with the back propagation algorithm.

WinNN has a user friendly interface written in Visual Basic and uses a FORTRAN DLL for fast network calculations. The combinations of speed and useful friendly interface makes this program attractive for the beginner and advanced Neural Network user.

Introduction
Getting Started
The NN Control Panel (NNCPL)
ToolBar
Menu
Windows
Examples
Revisions
Registration
Other Great Shareware

Introduction

Windows Neural Networks (WinNN) is a windows based Neural Network (NN) simulator with back-propagation learning. Following is a brief description of feed forward NNs that helps understand the way WinNN calculates and adjust the weights.

A neural network in its basic form is composed of several layers of neurons; an input layer, one or more hidden layers and an output layer. Each layer of neurons receives its input from the previous layer or from the network input. The output of each neuron feeds the next layer or the output of the network. This is illustrated in figure-1 which shows a 3 layers NN. The first layer is an input layer that distributes the inputs to the hidden layer and does not have any activation function.
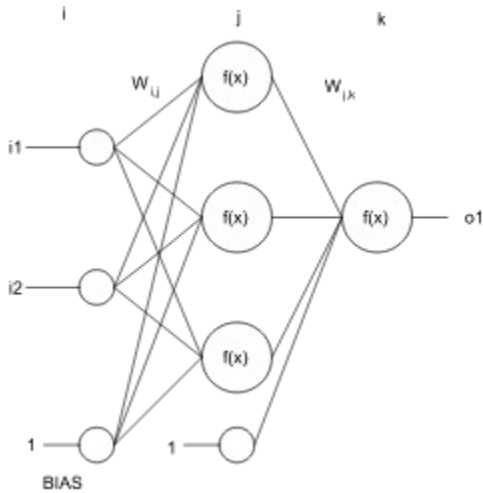

Figure - 1 A simple 2x3x1 NN. The lines connecting the neurons represent the weights. Also shown is the bias nodes that are used to shift the neuron transfer function and improves the network performance.

Mathematically the network computes:

1. The output of the hidden layer (treating the bias as another input):

$$h(j)=\text{Sum}(w(i,j)*i(i),\ i{=}1,\ 3)$$
$$s(j)=f(h(j))$$

2. For the output layer calculate:

$$h'(k)=\text{Sum}(w'(j,k)*s(j),\ j{=}1,\ 3)$$

$$O(k)=f(h'(k))$$

where:

$i(i)$ - are the network inputs.

$O(k)$ - are the network inputs.

$W(i,j)$ - represents the weight connecting neutron i in layer 1 to neutron j in layer 2.

$W'(j,k)$ - represents the weight connecting neutron j in layer 2 to neutron k in layer 3.

f(*x*) - is the neuron transfer function. for example a sigmoid:

$$f(x)=1/(1+\exp(-x))$$

Training such a network involve using a data base of examples which are values for the input and output of the NN. The NN would learn by adjusting the weights to minimize the error of the outputs. The error function is the objective of the minimization procedure and defined as:

$$\text{RMS Error}=\text{Sum}(\text{Sum}((t(p,k)-O(p,k))^2, k=1, K_{max}), p=1, P_{max})$$

 where:

O(*p,k*) - is the NN output *k* for pattern *p*.

t(*p,k*) - is the output training pattern *p* for output *k*.

WinNN uses a simple backprogation algorithm to adjust the weights, this algorithm is an iterative one. WinNN trains in BATCH mode with a variable EPOCH length. That is it sums the weight adjacments over all the training patterns in an epoch and then adjusts the weights.

Getting Started

WinNN is a MDI application (Multiple Document Interface). When WinNN loads the Neural Network Control Panel (NNCPL) is displayed as a maximized form. Follow these steps for a fast start.

1. **Prepare a pattern input file:** This file contains the input and output training patterns for WinNN. The first line contains 3 numbers:

      1. The number of training sets in the file.

      2. The number of   Inputs.

      3. The number of Outputs.

The next lines will include the inputs and outputs patters for each node separated by one or more spaces. For example the XOR training file will look like:

4 2 1

0 0   0

0 1   1

1 0   1

1 1   0

You can create this file using windows notepad and save it with   the extension .PAT

2. **Set the Net Size:** This is done in the NNCPL in the frame labeled "Net Size" :

      1. Select the number of layers (including input and output layers).

      2. Select each layer size by selecting the layer from the list box, type the new size and then click on the change button, repeat for all layers as needed.

      3. The neuron activation function for each layer can also be changed by selecting the desired function from the List Box labeled neuron func.

3. **Open the pattern file:** select the Open Patterns command form the FILE menu or click on the patterns box in the Files frame.

4. **Open a test file:** If you have a test file with patterns, you can choose to open it by using the FILE menu command or clicking on the test box in the Files frame

5. **Normalize the data:** if your data has large number that are out of the effective range of the neurons inputs or output, you should normalize your data. WinNN checks the pattern file and will advise you to do so if numbers out of range are detected. The normalization command is in the DATA menu.

6. **Start training:** click the toolbar play button ▣. To stop click the tool bar ▣.

The error in the RMS error box should now decrease as the training progress. When all patterns have an error smaller than the target error the training will stop.

7. **Save the network**: first save the weights by choosing Save Weights As from the File menu. The Save Net toolbar icon ▣ will appear, use it to save the network.

Control Panel

The NN Control Panel (NNCPL) controls the network settings and displays important information about you network training. The NNCPL is devided to groups of controls each surounded by a frame:

Net Size
Files
Learning Parameters
Weights
Training Results
Epoch Parameters

Net Size Frame

The controls in this frame allows you to set the network Size.

<u>Layers:</u> sets the total number of neurons layers including input and output layers, the input layer nodes are not neurons and used only to distribute the weights. the number of layers can be between 2 to 5.

<u>Layer Size:</u> to change layer size, first select a layer from the list box, type the new size and click on the Change button .   Repeat for other layers as needed. Layer Size is limited by the amount of memory that can be allocated for all the arrays in WinNN.

<u>Neuron func:</u> to change the neuron activation function for a layer, select the layer first and then change the activation function.   There is no need to click on the change button.

The function available are (T is the temperature):

Linear:

$$f(x,T)=xT$$

Sigmoid:

$$f(x,T)=1/(1+exp(-xT))$$

Hyperbolic Tan:

$$f(x,T)=tanh(xT)$$

Note: If your copy of   WinNN is not registered there is a limit to the total number of weights you can use with WinNN.

Files Frame

The file frame is used to opens and display the files that are being used:

Pattern File (extension .PAT): The file containing the training patterns. File Format

Weight File (extension .WGT): The file containing the trained network weights.

Test File (extension .TST): This file has the same format as the pattern file and contains test patterns that are not in the training set.

All files are ASCII files and can be edited by a simple text editor like notepad.

Learning Parameters Frame

This frame controls the way WinNN Train on the pattern data:

Eta and alpha relates directly to the backpropagation learning algorithm: where the new weights are a function of the derivatives and the previous weights for example:

$$d\textit{WW}=\eta d\textit{W}(i,j,t)+\alpha\textrm{Oldd}\textit{WW}(i,j,t\textrm{-}1)$$

$$\textit{W}(i,j,t+1)=\textit{W}(i,j,t)+d\textit{W}\textrm{W}$$

$$\textrm{Oldd}\textit{WW}(i,j,t)=\textrm{dWW}$$

Eta (η): is the learning parameter.

Alpha(α): The momentum.

Input Noise: Will add a random input noise of {The value} to each input node. Training with noise makes the trained network less sensitive to changes in the input values. and can help avoid local minima.

Weight Noise: Will add a random input noise of {The value} to each weight, This noise makes the network weights constantly "shake" as the training progresses. The purpose of this noise is to help the network jump out form a gradient direction which leads to a local minima in the weight surface.

Temp: Changes the temperature of the neuron function. The temperature is a multiplier of the activation argument.

in a sigmoid for example:

$$f(x,T)=1/(1+\textit{exp}(\textrm{-x*T}))$$

Changing the temperature sometimes makes the learning process faster, in most cases best results are obtained with the default value of 1.

Iter/calc: This number determines the number of Iterations per DLL calculation. WinNN training calculations are done by the WINNN.DLL, this .DLL is much faster than the WinNN VB. interface. By setting this number high the DLL will perform more iteration each time it is called, this will yield more CPU utilization for WinNN and less for other applications.

** To have WinNN work in the background and consume small amounts of CPU time use a small number (1 to 5) for the Iter/Calc parameter.

Clip Patterns: Allows to train only on patterns that have a larger error than the target error. The patterns that are trained are patterns that conform to:

$$\textrm{Pattern error} > \{\textrm{Clip Patterns value}\}*\textrm{Target error}$$

This feature is very useful when the percent of Good Patterns(pattern error < target error) gets

between 90%-100% and the network RMS error is much smaller than the target error.   Setting this parameter to 1 will training the only on the "bad" patterns and the RMS. error might increase but the number of Good Patterns will also increase.

Weights Frame

This frame sets and displays information about the network weights. The initial weights are selected randomly, the Min and Max text boxes can set the limits of the random number generator. The actual Min and Max values are also shown and will update when you stop or start the training.

Training Results Frame

This Frame shows the training results and allows the user to enter the training Target Error.

RMS Error: the training error is measured by squaring the difference between the network and training pattern desired output and summing over all outputs and all training patterns.

Change: is the difference between the current RMS error to the previous iteration value.

Iterations: shows how many times the network was trained on all the data points in the .PAT file. This number will increment in steps determined by Iter/Calc parameter.

Good Pats: the percent number of patterns with error less or equal to the target error.

Target Error: set the training end criteria. The training will stop when all patterns have an error that is less or equal to the target error, in other words when Good Pates=100.

Epoch Parameters Frame

Epoch Length: Sets the number of patterns to be trained before the weights are updated.   If for example you have 35 training patterns and the epoch length is 10 the weights are updated as follows.

Train patterns 01-10, update weights

Train patterns 11-20, update weights

Train patterns 21-30, update weights

Train patterns 31-35, update weights

You can set the epoch length to 1 to simulate CONTINUES training, the weights are updated after every sample. Setting the epoch length to the number of training patterns will simulate BATCH mode where the weights are updated once for each pass over all the training patterns. The last mode tends to yield faster training time.

If you want to yield more time to other applications, reduce the epoch length. However using epoch length smaller than the number of patterns requires WinNN to add an error calculation for each pass of all patterns this will slightly reduce the training speed.

To set the epoch length to its default value set the epoch length to zero. A message box will appear and the epoch length will be set to the total number of training patterns.

Random Sampling: Enabling this option will choose the training patterns randomly within each epoch. When this option is disabled (the default) the training patterns are selected sequentially. Using this option requires WinNN to add an error calculation for each pass of all patterns this will slightly reduce the training speed.

The ToolBar

The toolbar allows quick access to most used commands in WinNN. Some of the toolbar buttons activate commands that are not available in the menu. When the cursor is above an enabled toolbar button the status line in the bottom of the NNCPL will give a short description of the button function.

Creates a new network and resets all parameters.

Opens a saved network file. A network file will load the training patters, the weights, all the training parameters and will restore the desktop (order of all WinNN windows).

Saves (or Save As) the network, When you start WinNN this button is hidden, to make it active you must first load a pattern file, train the network and save the weights. The Save Net command will save all the training parameters and the position of all the windows in WinNN. When you Open the network file again you will get the same desktop layout you saved and all the network parameters are restored.

Start training the network, To enable this button; load a pattern file.

Stops the training.

Runs the trained network, This button will run the trained network on all the training patterns. The results are displayed in a list box, The results are:

{pattern no} {Network Output} {Target Output} ....   {} {} {} {RMS. Error} { + or -}

The + indicated that the pattern error is less or equal to the target error. The data is displayesd in the Report Window.

Test the trained network, This button has the similar function to the

button except the input and output patters are taken from the test file.

Plots the network weight distribution. The weight distribution can give information on the quality of the weights obtained. Normally for a network with the same neuron activation function in all layers, the distribution should be Normal. The information is displayed in the Weight Plot Window

Plots the network output against the target outputs form the training set. The information is displayed in the Output Plot Window

Menu

File
Edit
Data
View
Setup
Window
Help

File Menu

New Net

Creates a new network and resets all parameters. same as the toolbar .

Open Net

Opens a saved network file. A network file will load the training patters, the weights, all the training parameters and will restore the desktop (order all WinNN windows).same as the toolbar . (looks for extension .NET)

Open Patterns

Open a pattern file (looks for extension .PAT).

Open Test

Opens a test file, file format is same as the pattern file (looks for extension .TST).

Open Weights

Opens a weight file that was saved by the Save /As Weight command (looks for extension .WGT)

Edit Menu

This menu containes commands to enable <u>DDE</u> links to other applications.

Paste Pats (Linked)

Pastes patterns from the clipboard to WinNN. The data must be copied from an application that supports DDE (EXCEL, WORD). The pasted data should contain the input and output patterns of the network. The pasted data is limited to 32K (about 6000 numbers)

Paste Tst Pats (Linked)

Pasts the test patterns from the clipboard.   The pasted data is limited to 32K (about 6000 numbers)

Copy Pat Outputs

Copies the network training outputs to the clipboard. You can then paste link the data to another application. The copied data is limited to 32K (about 6000 numbers)

Copy Tst Outputs

Copies the network test outputs to the clipboard. You can then paste link the data to another application. The copied data is limited to 32K (about 6000 numbers)

Update Client

Updates the outputs pasted in a client application

Dynamic Data Exchange (DDE)

The Edit menu contains several commands that allows WinNN to send and receive data by a Dynamic Data Exchange (DDE) links with other applications. An example will be using data in EXCEL spreadsheet. To create a DDE linked network with EXCEL:

1. Run WinNN and EXCEL.

2. In WinNN, set the desired network size.

3. In EXCEL, load the data in columns, one columns for each input node and one for each output node. (like the pattern file (.PAT) format).

4 In EXCEL, select all   pattern columns (network inputs and outputs) and use the Copy command from the Edit menu to copy the data to the clipboard.

5. In WinNN select the Paste Pats command from the Edit menu. The data will be pasted and WinNN is ready to be trained.

Paste network outputs to EXCEL

6. Select the Copy Pat Outputs from the Edit menu in WinNN.

7. In EXCEL select the   columns where you want the data pasted. Select the Paste Link command from the Edit menu.

8. The inputs and outputs are now linked to WinNN. Any change in the input data in EXCEL will change the input patterns in WinNN and will cause the output data pasted in EXCEL to automatically updated.

Saving and loading a DDE linked network.

To save a network with DDE links, first save the weights and then save the network using the Save Net As command from the File menu. Loading a DDE linked network is done the normal way by clicking on the Open Net toolbar. WinNN will lunch the application providing the data.

Data Menu

Normalize

The normalize command will open the normalize form, this allows to re-normalize the pattern and test files to the effective neuron input or output values. This function is useful if the pattern values are in a different range than the useful effective range of the neuron activation value. For example for sigmoide output nodes can only have values between 0 to 1 if your patterns has output values between 1 and 10 the WinNN can not train on your data. The input to a sigmoide should be between -2 to 2 other values will saturate the neuron causing it to output 0 or 1 all the time.

Once the data is normalized, the plots and the network report will show the de-normalized values. However the errors shown are of the normalized data.

When the normalize form is open you will be presented with the minimum and maximum values of the inputs and output training patterns. The text box will contain the effective input and output ranges for your net. You can change these values if you like and then click the **NORMALIZE** button. The data will be normalized, and the new min. and max. values will be presents. The normalization can be linear of log-linear.

Linear: {New Value}=A*{Old Value}+b}

Log:　　{New Value}=A*ln{Old Value}+b}

The log normalization is user delectable and will be enabled only if there are no zeros in the patterns. You should use the log normalization of your data changes over a wide range. Linear normalization will divide the data by the max value this can cause the minimum data point to be very low and the network will not be sensitive to this input or output.

You can use the **Normalize** check box to select if the input or output will be normalized or not.

Normalization Method

Global: This option uses one set of normalization factors A and B for all inputs and another for all outputs..

By Node: Every input/output will have it's own set of normalization factors A and B. This option is useful when there is a mix of inputs with various ranges and is the default

RESET　reloads the training and test patterns with the original values.

View Menu

View Weights

Will Open the Report windows and display the weights layer by layer. The first number is the weight from node 1 in layer 1 to neuron 1 in layer 2. for example for a 2x2x1 network the text format is:

9   <----- Total no. of weights

2   <----- Weights from layer 1 to 2   (n-1 to n)

(1,1) (2,1) (3,1)   <----- input layer to neuron 1 in layer 2 (hidden layer)

(1,2) (2,2) (3,2)   <----- input layer to neuron 2 in layer 2 (hidden layer)

3 <----- Weights from layer 2 to 3 (n-1 to n)

(1,1) (2,1) (3,1)   <----- hidden layer to neuron 1 in layer 3 (output layer)

As you noticed there is an extra node feeding layers 2 and layer 3, this is the bias node that has a weight too.

The data is displayed in the Report Window.

View Weights Delta

Similar to View Weights, but will show the weights delta which refers to the weight increments in the last training set.

Setup Menu

Editor: Allow you to use another text editor to edit the pattern, weight and test files. The default editor is Notepad. Any change made is recorded in the file winnn.ini (in /windows) and will be used any time you run WinNN. If you choose a DOS editor you can define a windows PIF file to prevent from shelling to DOS. Use the PIF file name as the editor file name and make sure it is in your path.

Search local directory first: When the network is saved, the full file names (path+name)   for the patterns (.PAT),   weights (.WGT) and test (.TST) files are saved. When loading a .NET file, if this option is not enabled WinNN will search the file by its full name including path. If this option is enabled (default) WinNN will first look for the file in the current directory (the one with the .NET file) and if it failes to find the file it will look in the full path. This option   is useful when copying files from one system to another and the files in the other machine are in a diffrent location.

Window Menu

Use the commands in this windows to navigate between the different windows of WinNN.

Help Menu

Help

Will run this help file

<u>Register</u>

Once you obtain a user code you can register your copy of WinNN by using this command

<u>About</u>

Shows WinNN version and information about free memory.

Windows

[Plot Outputs](#)
[Plot Settings](#)
[Plot Weights](#)
[Report Form](#)

Plot Outputs

This windows is displayed when the plot output toolbar ⊠ is pressed. The target outputs and calculated network outputs are plotted. The plot can be used to visualize and examine how well the network learned the data.

Use the **Plot Type** List box to select between
- Net outputs for training set.
- Error of trained data (normalized data).
- RMS error as a function of the number of iteration.
- Net outputs for the test patterns. This can give some information on how the network generalize when running on data not in the training set.

If the data was normalized the plots will show the un-normalized data.

The plot can be copied the clipboard in a Windows MetaFile format and can be printed with the PRINT command in the File Menu.

The intervals (in seconds) in which the plot is updated can be set and will be written to WINNN.INI this setting is common to all the .NET files.

Plot Settings

This window pops up when the right mouse button is clicked on the plot window. The allows various settings of the plot to be changed.

Title: Changes/adds plot title.

X-Label: Changes/adds X axis label.

Y-Label: Changes/adds Y axis label.

Type: Changes the plot type, Line, Log Line, Line+symbols, Symbols, 2D bar, 3D bar.

X-Grid: Adds an X axis grid.

Y-Grid: Adds Y axis grid.

Color for each curve: When enabled each plotted curve would have a different color.

The plot settings can be changed for the patterns, errors and test plots and are saved in the .NET file.

Plot Weights

This windows is displayed when the plot output toolbar is pressed. The network weight distribution is plotted. The weight distribution calculated by bining the weights to 10 equaly spaced bins, the number of weights in each bin is plotted vs. the bin (weight) value. The weight distribution can give information on the quality of the weights obtained. Normally for a network with the same neuron activation function in all layers, the distribution should be Normal.

The plot can be copied the clipboard in a Windows MetaFile format and can be printed with the PRINT command in the File Menu.

The intervals (in seconds) in which the plot is updated can be set and will be written to WINNN.INI this setting is common to all the .NET files.

Report Window

The report windows servers as a text display window for several functions:

- View Weights

- View Weight Deltas

- Run Network (toolbar)
- Test Network (toolbar)

This window can display up to 32 K of text, the text can be copied to the clipboard. However for problems with large number of training patterns the results of running the network might yield more than 32 K of text data. The current solution to this problem is to use the *Save And Edit* command from the Report File menu. This command will save the data to a file and then invoke the editor to view the data. The default editor is notepad and is also limited to 32 K. You can change the editor form the Setup menu.

If the data was normalized the report will show the un-normalized data. The errors shown are of the normalized data.

Examples

Several examples are provided with WinNN to help understand how the program works. You can load the examples by selecting the Open Net command from the File menu or by clicking on the open net toolbar icon. The examples contain pattern files test files and weight files. All the examples includes the weights of a trained network.

The examples will load the desktop. This desktop was save on 1024x768 display resolution and therefore might appear big on lower resolutions. You can use the scroll bars to navigate in the desktop.

XOR2

XOR2N

XOR7

SINCOS

LOGISTIC

XOR2

This examples trains a 2x2x1 network to simulate a XOR function. The network is trained on a set of data and then tested on another set where the inputs are not exactly 0 and 1. clicking on the Test Net toolbar icon shows that the network is not calculating two of the test inputs at the required accuracy. To overcome this problem go to example XOR2N You can retrain the network by clicking on the randomize button and the on the train toolbar icon. You should see that if you repeat that process several times, in some cases the training will get stack in a local minimum and the error is not being reduced.

XOR2N

This demonstrated how to use the input noise parameter to make you network more robust. The XOR2N problem is the same as XOR2 except that the network was trained with 0.2 input noise. Clicking on the Test Net shows that in this case the test data is calculated accurately by the network.

XOR7

This problem demonstrate the ability of WinNN to work on larger problems. The XOR7 problem trains the NN to function as a 7 bit XOR or as a parity detector. The output of the NN should be one only when the number of input bit that are set to one is odd.

The NN was trained only on 100 patterns out of the 128 possibilities, the rest of the patterns were used as a test set to see if the NN can generalize the problem to data it never saw. This case shows that for the 7 bit XOR problem the NN can learn the data but can not effectively generalize.

SINCOS

This examples trains an NN to simulate a continuous function. The network chosen is a small one 1x4x2. Although this network is small, it can learn to simulate the sin and cosine function accurately. This example shows that an NN can be used effectively to interpolate multidimensional continuous functions.

LOGISTIC

This example demonstrates WinNN Dynamic Data Exchange (DDE) capability. To run the example EXCEL has to be installed on your system. The training data is stored in the file LOGISTIC.XLS. When you load LOGISTIC.NET WinNN will lunch EXCEL and load the pattern data. The output of the network will also be linked to EXCEL.

In this example WinNN learns to "predict" the logistic map equation given by:

$$x(n+1)=4*x(n)*(1-X(n))$$

You can try changing the first input number in column A6 in the EXCEL spreadsheet to see the effect on the network output.

Revisions

Version 0.90 February 94
Initial Release.

<u>Version 0.91 March 94</u>
- Many improvements data normalization.
- Add plot settings. and improve plotting.
- Enhancements in the performance of WINNN.DLL
- Major changes to the .NET file. Version 0.91 can not read version 0.9 net file. To overcome, redefine the network size and load the patterns, weight and test files.

<u>Version 0.92 April 94</u>

- Add DDE links of inputs and outputs to other applications.
- Add plot of RMS error as a function of the number of iterations.
- Various bug fixes.

<u>Version 0.93 May 94</u>

- This version corrects several bug found in version 0.92.
- Replaces versions 0.92a, 0.92b and 092c.

Other Great Shareware

Other Great Shareware is also available:

Digitize : An UnGraphing program for windows 3.1

WinFit : A non linear least squares fitting program for windows 3.1

Digitize

Description

Digitize does the reverse job of a plotting program (Un-Graphing), it can import a scanned X,Y plot and digitize it to end up with a text file containing the X,Y points.

Digitize is a very successful sharware program, some of its capabilities are not found in other commercial packages that costs ten time the registration fee of Digitize.

Look for the file digi138.zip on ftp.cica.indiana.edu

Some of   features and capabilities of Digitize

• Load bitmap pictures from a file (BMP,PCX) or paste from the clipboard.

• Define plot axis for any type of plot (linear, Semilog in X or Y, and log-log)

• Digitize manually or automatically.

• ZOOM window (with variable magnification) lets you see every pixel.

• Display the ERROR of the digitized points.

• AUTO TRACER can digitize the curve with high accuracy:

   • Two Auto Trace algorithms and a SCAN mode help Digitize various type of plots.

   • SCAN Mode to digitize any type of plot/picture.

   • The auto tracer can digitize multiple INTERSECTING curves with the help of simple blockers with adjustable size.

• ERASER to clean the picture.

• View and digitize the image in the scanned resolution, or reduce and stretch the image to fit your window space.

• SAMPLE option allows sample the digitized points with different X spacing.

• Save the digitized points to a file or copy to the clipboard to paste directly to other applications.

• Supports several text formats for the output.

• Detailed on line help.

• TOOLBAR with most useful functions helps working faster and easier.

• Save your setting configuration to DIGITIZE.INI

WinFit

A general purpose non linear weighted least squares fitting program for windows 3.1.

This program was uploaded to ftp.cica.indiana.edu as wfit14.zip.

FEATURES

- The program uses Levenberg-Marquardt fitting method.
- Reads a simple ASCII file, space or tab delimited set of X Y points with an optional Y error column.
- The data can be plotted with log axis options.
- Fitting algorithm is implemented in a DLL for faster code execution.
- There are some built in functions and the user-defined functions which can be saved.
- The program can generate weights that will improve fitting performance for some problems.
- This version can read up to 5000 data points and fit up to 27 parameters.
- The program provides a report file and the plot can be copied to the clipboard.
- The program will calculate and display the COVARIANCE and CURVATURE matrixes

Registration

If you use WinNN for more than 30 days (or you liked it much sooner) you should register by paying the registration fee. Please mail the payment to:

Yaron Danon
14 Beman Lane
Troy, NY 12180
(danony@rpi.edu)

A registered user will receive a user code that registers this version of WinNN and removes the connections limit and the nag screen. This user code will also work with future shareware releases of   WinNN. New versions will be uploaded to ftp.cica.indiana.edu, to upgrade, all you have to do is download a new version and register it using your code. When you register you will also receive the latest version of WinNN on a disk and be eligible for technical support by E-mail.

To register print and fill this form or use the file REGISTER.WRI

Register: ___ Personal Copies at $35 each.

                        ___ Commercial Copies at $50 each.

Total enclosed $ ____

Please send check or money orders in **US dollars drawn in a US bank**, no credit card orders.

Name:

_____

Address:

_____

_____

_____

_____

Internet address: _____

User Name: _____

            (First and Last name for personal registration users)