

The authors welcome your comments on the program and this manual.
Please send your comments to:

aiNet,
Trubarjeva 42
SI-3000 Celje
Slovenia
EUROPE
e-mail: **AINET@SIOL.NET**

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form by any means, electronic, mechanical, photocopying, recording, or otherwise, without permission of the authors.

A neural network application for Windows environment

Every endeavor has been made in the preparation of this manual and the software. The authors make no warranty of any kind, expressed or implied, with regard to the program or to the documentation contained in herein. The authors shall not be liable in any way, even for incidental or consequential damages in connection with performance, or any kind of use of the program.

Version 1.24, August 1997

© aiNet, 1995-97

Acknowledgment

We express special thanks to Mr. Mark Charles for his excellent manual review.

Microsoft Windows, Microsoft Word, Excel, Windows 95, Visual Basic
are registered trademarks of Microsoft Corporation,

Borland C++ and Quatro Pro
are registered trademarks of Borland International, Inc.

Preface

In the beginning of 1992 Iztok Perus, one of our team members, began using neural networks. He tried to solve some simple problems in the field of civil engineering and the results were very promising. He also encouraged some of us to try neural nets. We, like probably everybody else, also started with simple problems and we solved them successfully. This stimulated us and we actually began to compete; the winner was the one whose neural net took the longest time to learn☺. Unfortunately this game was over very soon. We found a problem, which seemed to be simple enough, though a little noisy. At least we thought so. But we were wrong, very wrong. We ran our neural net programs with all possible combinations of neurons in many different layers. Several weeks were spent, without any results. We gave up; the problem remained unsolved and we lost our faith in neural networks. (We must confess our knowledge about neural nets at that time was pretty poor. Some expert would probably succeed in solving it.)

A few months later Iztok got an article from Prof. Grabec, where some new ideas about neural networks were presented. He added some improvements, programmed his computer and tried out the new neural network on some simple problems. Everything worked fine. Finally, he also tried to solve our “unsolved problem” and he succeeded. Moreover, he solved it in one hour.

Naturally, we did not believe him. Later he convinced us. I remember we were somehow disappointed because everything we wanted to solve with the ECA[#] (as we named the new neural net at that time) had been solved so quickly - it was a matter of a few minutes or even a few seconds.

The story repeated itself a few months later. Iztok was visiting a university in Italy, where he met a team, who were also concerned with neural networks. They had a problem and after they had made several experiments with a number of hidden neurons, the final learning phase took about 30 hours. The results were good, but Iztok achieved better results within half an hour, although he had never heard of the problem before. They did not trust his results (Just like we did not in example above.) The awareness that something can be done a couple of times faster than it was done before is always frustrating, especially if the new way is much simpler, too.

In the 1994 we came up with an idea of writing a general purpose neural network application. However, having an idea is one thing and facing reality another. We could find no one to finance the development of such an application. Finally, we decided to try it on our own and here it is: **aiNet**. We wrote this application in our spare time and we became more and more dedicated to aiNet.

Thus, version 1.0 of aiNet is complete. But how do we find customers? We have no money to finance a marketing campaign and we even do not know if the users will like our application.

[#] ECA stands for “Estimator of Conditional Average”

Luckily, the Internet is there and we decided to put our application on it as a shareware. We really hope you will like it and register it. There are still so many ideas around and we want them to be built into aiNet, but our resources (time & money) are limited. By registering this application you will help us to continue our work and to continuously improve aiNet.

Ales Krajnc, January 1995

Introduction

aiNet is a very powerful and simple tool for solving the problems which are usually solved with artificial neural networks (ANN). The tests that we have run, have proved that the results obtained with aiNet are at least as good as the results obtained with other ANNs.

Here are some of aiNet's features.

- The major attribute that distinguishes aiNet from other ANNs is the analysis speed. Since aiNet uses an algorithm, which does not require any learning phase, the answers about prediction can be obtained almost immediately.
- There is also only one coefficient (penalty coefficient), which has a major effect on the results. If we neglect some aspects, we can claim that knowing the right value for this coefficient solves the entire problem. One would now probably expect that it is hard to determine the optimal value of the coefficient. On the contrary, it is quite easy! All you need to do is to try different values and finally select the most successful one. According to our results there is only one optimal value for the penalty coefficient. Fortunately, the results are not sensitive to the optimal value. This means that if you slightly change the penalty coefficient from the optimal value, the results will not change much (if at all). The optimal curve is usually shallow at the optimal point.
- Another very important feature is the ability to dynamically change the “knowledge base”. This means that new data may be added to the neural network (or old data removed), additional variables may be added (or old ones removed) and answers obtained right away -- there is no time consuming learning phase.
- Our experiments show us that noisy data is processed by aiNet particularly well. When the data is chaotic, and there is no possible solution, aiNet will indicate a data problem.
- aiNet provides a way to estimate the rate of error in the prediction. If the problem is smooth, i.e. without noise, then this error will represent an estimation for the error in the predicted result. If the data is noisy, then this will represent an estimation for the noise around the predicted result. This means that error estimation behaves locally.
- aiNet can cope with missing values in the data. In real life it is usually very difficult to find a perfectly assembled knowledge base -- there is always some data missing. aiNet handles missing data automatically without requiring representation of the missing data.
- aiNet's graphical user interface is simple to use. It is designed like a spreadsheet application for interface familiarization. Almost everything is only a mouse click away, menus are simple and there are also speed buttons.
- On-line help is provided and may be accessed in the usual Windows formats..
- Several charts are also available. They represent the most natural way to estimate how good the data set is, indicate visually if the problem may be generalized and what kind of results may be expected.

- With the new two parametric chart, the calculation results may be presented in 3D space.
- aiNet provides excellent links to other applications, especially spreadsheets. This means that results calculated in aiNet can be easily transferred to spreadsheets and vice versa.
- Along with aiNet, a development kit built around the aiNet DLL library is provided. This library includes the major aiNet functions needed to calculate predictions. This allows other applications to be developed using the aiNet DLL functions.

New Individual model vectors can be excluded from the model directly from the correlation chart or from model vector view.

New The DLL library has many new exiting functions. See Part 4 for more details.

Those are the features and advantages of aiNet. Like every product, aiNet also has some disadvantages. We found two of them (you will probably find more, we hope there would not be to many).

- aiNet is not suitable for all real time problems. When aiNet calculates a single prediction, it scans through all the data in the “knowledge base” which takes some time. On a PC 486/66 it takes about 0.1 second to compute one prediction based on a knowledge base with 1000 samples, each with 5 variables. This may present problems for some real time applications.
- aiNet does not generate fixed weights, which can be later used for prediction purposes. aiNet does compute weights, but the computing is done dynamically -- individually for each different prediction.

Manual Overview

This manual is divided into five parts:

- ◆ Part 1: Users Guide
- ◆ Part 2: Basics About Modeling with aiNet
- ◆ Part 3: Examples
- ◆ Part 4: aiNet DLL Library
- ◆ Appendix: Mathematical Description of aiNet

Part 1: “Users Guide” defines the fundamentals of aiNet. Firstly it deals with installation of aiNet. Later it defines data entry, how to save and load data from disc, how to perform analysis, which buttons to push and which menu commands to select. Examples are used to demonstrate aiNet’s features. The first example is the simplest possible and has besides educational, has no other practical meaning. The second one is slightly more complicated and has been selected to outline some features of aiNet. aiNet’s file format is also fully defined to enable further interface. At the end of Part 1, future additions and enhancements to aiNet are discussed.

Part 2: “Basics About Modeling with the aiNet” defines the modeling of problems. It explains major aiNet features in detail and shows how, why and when to use them. The explanation of these details is also based on simple examples.

Part 3: “Examples” deals with several problems from various areas of science. Every problem begins with an introduction, where the problem and goals are presented. The introduction is followed by the modeling section. In this section we explain how the data was modeled, which tools were used and what kinds of results were obtained. We always end with a comment section, where all possible additional information and important notes are given.

Part 4: “The aiNet DLL library” defines how applications may use aiNet’s prediction engine. First it explains in what order the aiNet DLL functions must be used. Later it advises how to program in C (C++) and Visual Basic and also shows each DLL function in detail.

Appendix: “Mathematical Description of the aiNet” was added to the manual to show the users the mathematical background of the method built into aiNet.

Manual Conventions

There are a few conventions used in the aiNet manual. We used different fonts for different types of meaning. These are defined as follows:

`Monospaced type` This font represents user entries, or an on-screen text.

Italics Italics are used to emphasize certain words, menu choices and indicate aiNet terms.

Keycap This font represents a particular key to be pressed -- for example, "Press ***Del*** to erase."

ALL CAPS All caps are used to represent disk directories and file names.

Menu|Choice Rather than using the lengthy phrase "choose the New command from the File menu," this manual uses convention "choose *File|New* command".

Contacting the Authors

Currently, we have limited capabilities for a customer support. We may be contacted either by post, or via e-mail. The addresses are listed below:

Post address: **aiNet,**
 Trubarjeva 42
 SI-3000 Celje
 Slovenia
 Europe

NEW E-MAIL address: **AINET@SIOL.NET**

We guarantee that all registered users will receive answers to their questions and that their proposals will be taken into account. The same guarantee is not valid for unregistered users, although they might receive answers from us.

We are continuously trying to improve our customer support and offer better service as the number of registered users increases.

NEW We may also be contacted by fax on fax (++386 63 49 00 681) and telephone (++386 63 49 00 680), but please reserve this for urgent inquiries only.

NEW Recently we have set up a web site. We suggest that you visit it from time to time to see if anything is new. The site URL is <http://www.ainet-sp.si/>.

**USER'S
MANUAL
Part 1:
Users Guide**

Table of Contents - Part 1: Users Guide

1. Getting started	1
1.1 Installing aiNet	1
1.1.1 Hardware and Software Requirements	1
1.1.2 Installation	1
1.2 Solve the first Problem	2
1.2.1 About the XOR Problem	2
1.2.2 Modeling XOR With aiNet	2
1.2.3 Comments	7
2. aiNet in Details	8
2.1 Different Views	8
2.2 aiNet Menu	10
2.2.1 Static Menus	11
2.2.2 Dynamic Menus	14
2.2.3 Three Most Important Commands	21
2.3 A more complex Problem	25
2.3.1 The Hole in a Square Problem	25
2.3.2 Generating Model Vectors	25
2.3.3 Modeling with aiNet	26
2.3.4 A Hole in a Square Problem with Noisy Data	29
2.3.5 Some Possible Improvements	30
2.3.6 Final Comment	30
2.4 How Can I Use My Data with aiNet ?	30
2.5 Printing and Copying	34
2.5.1 The Copy Command in Different Views	34
2.5.2 Two Parametric Analysis - 3D Surface Chart	36
2.6 aiNet's File Formats	39
2.6.1 The CSV File Format	39
2.6.2 The PRD File Format	40
2.6.3 The AIN File Format	41
3. aiNet in the Future	42
3.1 The next releases of aiNET	42
3.2 The "More Hazy" Future	42
3.3 You Can Help Us	42
4. All About Registration	43
4.1 What are the Benefits of Registration ?	43
4.2 How to Register	43

4.2.1	Registering the aiNet DLL library	44
4.3	Disclaimer of Warranty	44

Chapter 1

1. Getting started

This Chapter firstly defines the installation of aiNet. Afterwards it will be used it to solve a very simple problem. This introduction is preparation for the more complex examples in the next chapter.

1.1 Installing aiNet

Experienced Windows users are probably able to install aiNet and run the application without assistance. For those less familiar with this process, the following instructions should be followed.

1.1.1 Hardware and Software Requirements

aiNet requires the following minimal configuration:

- a PC with the 386 processor
- a minimum of 4 MB of RAM
- about 5 MB of hard disk space,
- Microsoft Windows95, Windows NT or as a minimum, Microsoft Windows 3.
- a VGA graphics card (aiNet does not support Hercules mono and EGA graphics, although Windows do),
- a mouse compatible
- on 386 and 486SX platform a math co-processor is highly recommended

Please note that later versions of aiNet may require different minimal configurations.

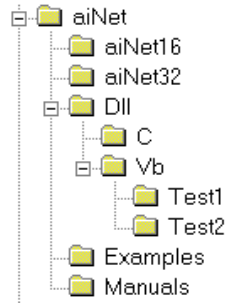
1.1.2 Installation

We supply aiNet in a compressed format, including all of the documentation and software. It is assumed that the application has been decompressed and is ready for installation.

Some of the files provided may already be on your PC, as they are supplied as standard with Borland compilers. Search your PC for an existing version 'BWCC32.DLL'. Move the latest version of this file to the WINDOWS\SYSTEM directory.

aiNet will also create a file called AINET.INI in the WINDOWS directory. The remainder of the files will be installed into a new directory on the disc.

Let us assume that the aiNet files are decompressed into the sub-directories as shown below. If you do not have such directory structure, we recommend you to decompress AINETXX.ZIP file again and do not forget to specify the -d option in the command line.



Correct aiNet sub-directory structure

1.2 Solve the first Problem

We shall now proceed to solve the first simple problem using aiNet. This will give some basic ideas of how aiNet may be used. The XOR problem was chosen for this presentation, mainly because several other neural networks also use it for their presentation.

1.2.1 About the XOR Problem

The XOR problem does not require a neural network to solve it, your biological network does this job much better. The XOR data is presented in a table form (see Table 1.1). The task is to verify that aiNet can compute (predict) correct results, after it has seen this data.

Table 1.1: XOR problem.

A (input)	B (input)	Result (A XOR B) (output)
1	1	0
1	0	1
0	1	1
0	0	0

1.2.2 Modeling XOR With aiNet

XOR will be modeled in several steps; data will be entered and then normalized. Test values will then be entered, and finally a prediction will be run and the data saved.

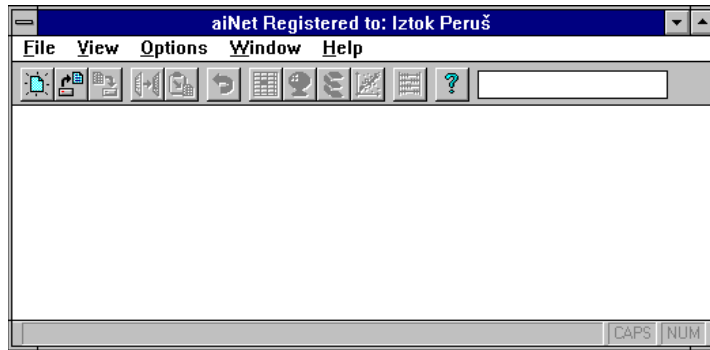


Figure 1.1

Entering the Data

Run aiNet. An empty window, with a menu and a tool bar appears on the screen (see Figure 1.1).

In the *File* menu select the command *New* and a dialog box will pop up. The dialog requires the number of model vectors (data sets) and the number of variables to be defined. There are 4 different sets (model vectors) and each model vector has 3 variables (value A, value B and the Result). Enter the numbers 4 and 3 and close dialog by pressing *OK* button (see Figure 1.2).

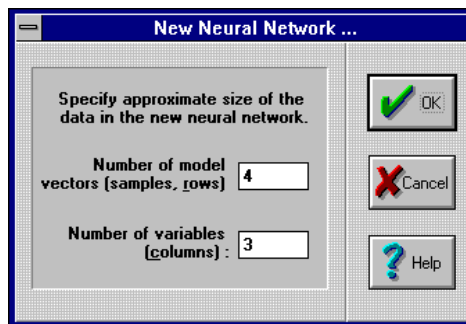


Figure 1.2

Now aiNet will ask for the Document type. Select either of the formats. At this stage, it does not matter which one is selected. The formats are fully defined later.

A grid of cells (4 rows and 3 columns) is now displayed. Type in the values shown in Table 1.2, so that the screen is as depicted in Figure 1.3.

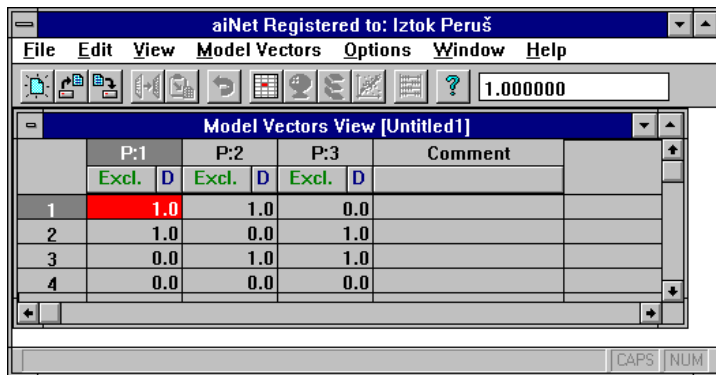


Figure 1.3

NOTE: When asked to confirm some values, answer with the *OK* button.

The data has now been entered and it is required to define the data as either input or output. The buttons above each column define the data type, which can be excluded, input or output. The default is 'excluded'. By clicking on the button you will change 'excluded' to 'input', 'input' to 'output' and 'output' back to 'excluded'. Change the defaults by pressing the appropriate button once or twice in the column headers (currently named as P:1, P:2, P:3). Set both P:1 and P:2 to *input* and then P:3 to *output*. (see Figure 1.4).

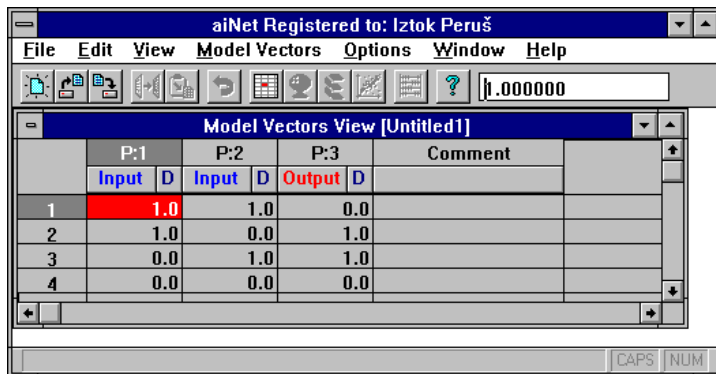


Figure 1.4

aiNet allows the variables (column headers) to be renamed. To do this, double-click on the variable name (P:1). Using the dialog box, change the variable name to "A" and then close the dialog box. Repeat this action for variables P:2 and P:3 and name them as "B" and "Result" (see Figure 1.5), respectively.

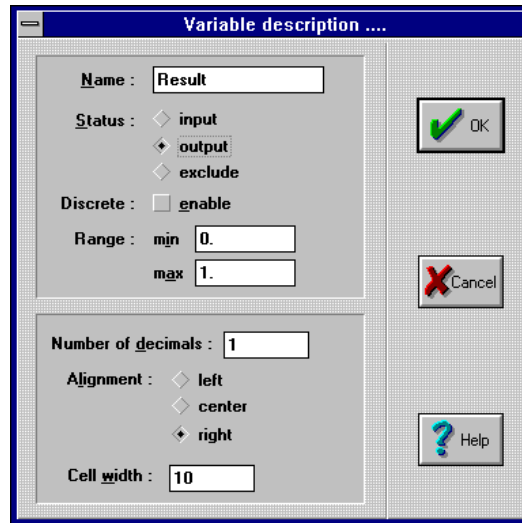


Figure 1.5

The model vectors have now been fully defined and we may proceed to the second step.

Normalizing the Model Vectors

This step is very short. Select *Model Vectors|Normalize+Lock* command and all model vectors will now be normalized (see Figure 1.6). Note that the numbers in the cells have changed the normalized values are displayed, instead the real ones. The cells are now locked and may not be edited.

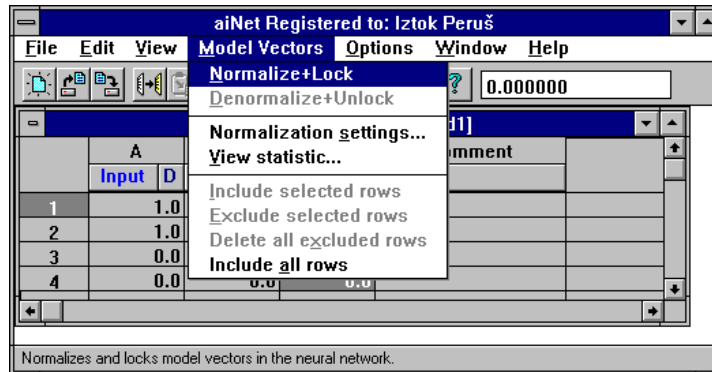


Figure 1.6

Running the Prediction

In this step we will see if aiNet is able to correctly predict the result from the normalized model vectors.

Select the *View|Prediction* command and a dialog box will pop up. It will ask how many rows are required in the prediction window. This actually means how many prediction vectors (sets) are required. Enter the number 5, for example (see Figure 1.7).

A new window with 5 rows will be displayed. This is called a “Prediction View”. Now test data may be entered into the *Prediction View*. Type in the numbers for input variables A and B as defined in Table 1.2. Note that data may not now be entered into the output variable cells (Result).

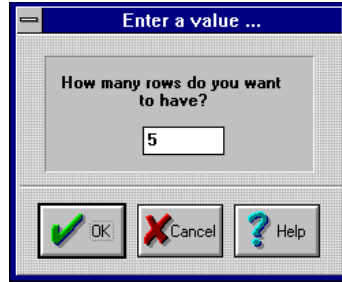


Figure 1.7

Table 1.2: XOR problem -test example.

A (input)	B (input)	Result (A XOR B) (output)
0.0	1.0	1.000
0.2	0.9	1.000
0.3	0.7	1.000
0.4	0.6	0.997
0.5	0.5	0.500

To calculate the results, select the *Prediction|Calculate Prediction* command. The results of the calculation will be presented in the third column. As shown in the table above, the results are displayed with three decimal digits, whereas the result in the *Prediction View* with has only one decimal digit.

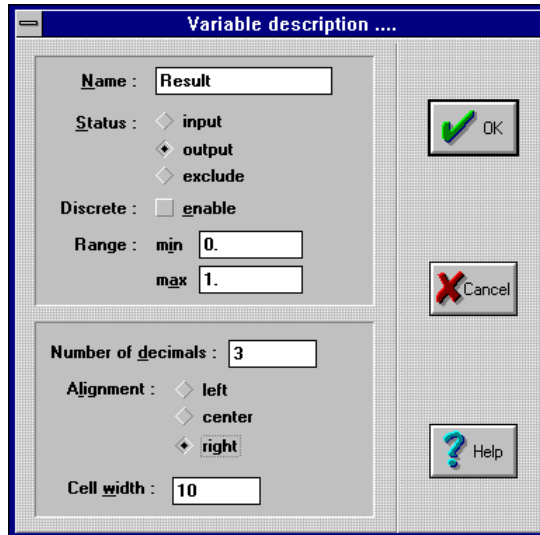


Figure 1.8

To change the number of decimals displayed, double-click on the `Result` output variable, enter number 3 beside the text “Number of decimals” in the dialog box (see Figure 1.8) and close it by pressing the *OK* button. Three decimals should appear in the R result column.

To experiment with new values, enter these into the input variables and then select the *Prediction|Calculate Prediction* command to display the new predictions. It’s easy, isn’t it?

Saving the Data

To save the data select the *File|Save* command. aiNet will display a Save As dialog, where a file name (`test1.csv`, see Figure 1.9) and a location (default is `c:\aiNet`, as can be seen from Figure 1.9) may be entered. A file format in the *Save File as Type* combo box may be selected. Two different file formats are available, an aiNet binary format and an ASCII format designed for data interchange.

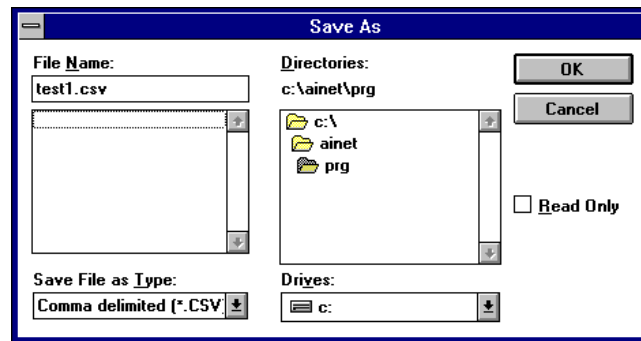


Figure 1.9

The ASCII format “Comma delimited (*.CSV)” stores all the model vectors and prediction vectors plus variable names. To use this format, the model vectors must be de-normalized first. This format is suitable for transferring information into other applications, like Microsoft Excel, Borland Quatro Pro, ... See also Chapter 2, aiNet File Formats.

The Binary format “AINET files (*.AIN)” store the data in the aiNet internal format. This format is not transferable, but stores additional information. For this format, the model vectors do not need to be de-normalized before saving. It is recommend that the binary format is used, unless the data is to be exchanged with some other application.

1.2.3 Comments

If we take a look at Table 1.2 with the prediction results, we can conclude that aiNet did the job well. In the first row we entered the prediction vector that is also present in the *Model Vectors View* and is part of the XOR problem’s knowledge base (model vectors base). In the second row we changed the prediction vector from the first row for 0.2, in the third row for 0.3, in the fourth for 0.4 and in the last row for 0.5. Although we considerably changed the second prediction vector according to the first one (each input variable for 0.2), aiNet calculated a correct result. Even when this is changed to 0.4, aiNet still calculates a good result.

Only the change of 0.5 puts the aiNet into the state of confusion. The last prediction vector { 0.5, 0.5 } lies exactly in the middle of any combination of the model vectors in the model vectors base. Thus, the aiNet gave us a logical result (0.5) for the fifth prediction vector, too.


Chapter 2


2. aiNet in Details


This chapter discusses further possibilities that aiNet offers the user. Because aiNet uses some specific commands, it may be worth reading through User's Manual, Part 2: Basics About Modeling with the aiNet, before continuing with this chapter. Although this is not necessary, it might lead to a better understanding of some of the topics presented here.


2.1 Different Views

The XOR problem demonstrated two 'views' possible using aiNet. These were *Model Vectors View* and *Prediction View*. aiNet has two additional views: *Error Distribution View* and *Charts View*. As the names of the views reveal, they present the problem in different ways. Views may be switched by selecting commands from the *View* menu, or by pressing appropriate buttons on the toolbar.

 *Model Vectors View* is a basic view, where all model vectors are presented in a spreadsheet form. The rows represent model vectors and the columns represent variables. Scroll bars allow large vector sets to be viewed. This is the default view and is automatically activated when a new neural network is created, or an old neural network is loaded from disk. This view is used to edit model vectors, to add new model vectors, or to remove old model vectors, etc. This view is also used to normalize and de-normalize model vectors.

 *Prediction View* is used for running predictions. This view is very similar to the *Model Vectors View* and allows similar actions to be taken. Inputs vectors may be altered and aiNet then be instructed to recalculate the output part. The Prediction view is therefore the most useful view.

 When the aiNet calculates the prediction it usually makes some mistakes -- the predictions are not 100% correct. Thanks to the special algorithms, aiNet can besides the prediction itself, also predict the ratio of the error in the prediction results. In order for aiNet to provide this information, a filtration and verification process must run over all model vectors. *Error Distribution View* shows the results of the filtration and verification. Like both of the views above, this view also uses spreadsheet-like window to present the results. This view is a Read only view.

 *Charts View* is an extension of the *Error Distribution View*, but it displays the results graphically rather than in tabloid format. This enables a quick estimation of the results visually. Three different charts are available. The first one is a correlation chart (see Figure 1.10). On the horizontal axis are output values from model vectors (correct values) and on the vertical axis are values calculated by the aiNet. If the calculated and the correct value for some model vector are the same (or almost the same), then the point of this model vector lies exactly on the diagonal line. In other words this means, closer the dots are to the diagonal, less error is present in the verification (red dots) or filtration (green dots).

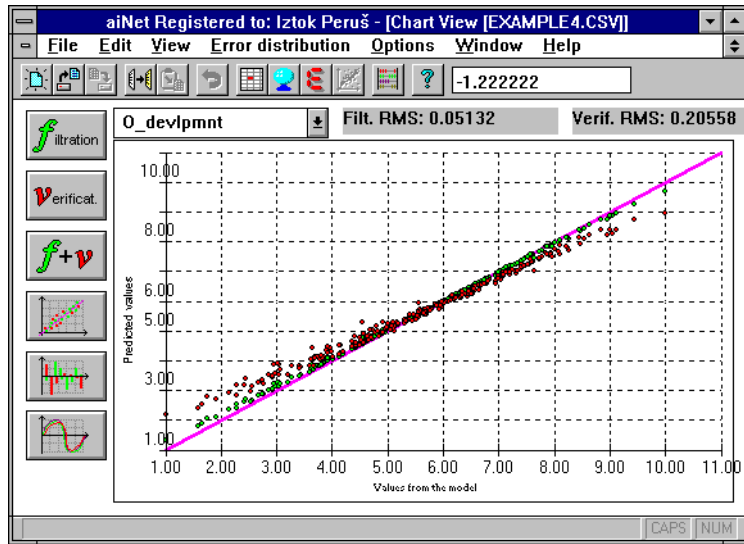


Figure 1.10

Sometimes users want to know, which are the model vectors that fail prediction either in filtration or in verification process.

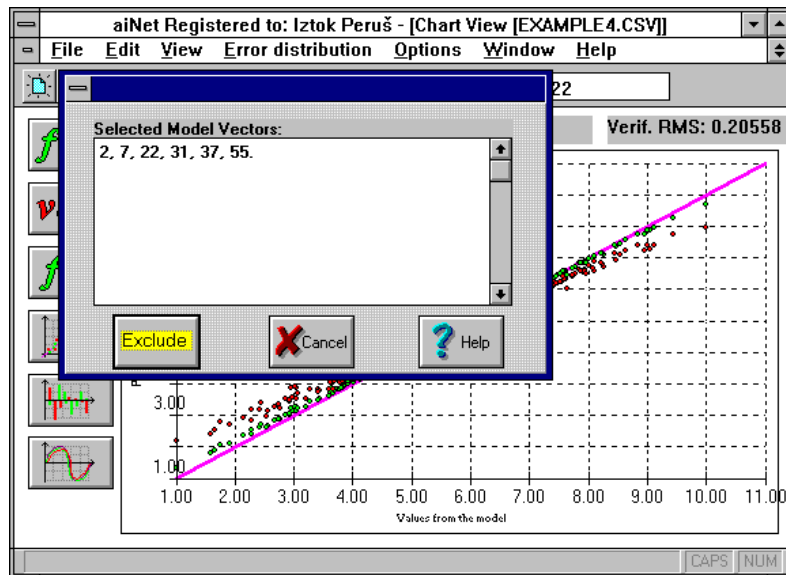


Figure 1.11

In order to select points of interest, drag the mouse, releasing the button at the appropriate location. An information window will pop up and give indices of the selected model vectors (see Figure 1.11). These indices are sequential numbers of the model vectors as they appear in the database. You are given a chance to exclude these model vectors - select the 'Exclude' button.

The second chart is a bar chart (see Figure 1.12). It shows the verification or filtration error of each model vector. The Model vectors are enumerated on the horizontal axis and the error is shown on the vertical axis.

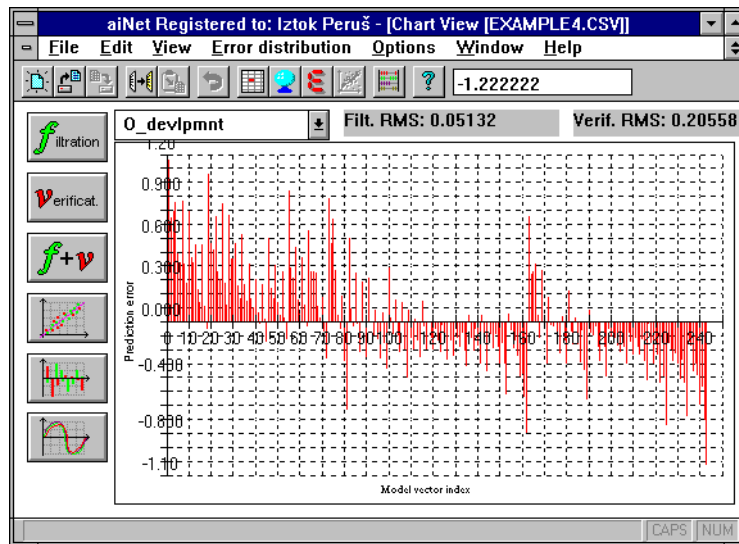


Figure 1.12

The third chart is similar to the bar chart. It shows the verification (red) or filtration (green) results of each model vector, both compared with the values in the model - actual values (black). Predicted model vectors are presented in sequential order, as they are written in the data base - input data file.

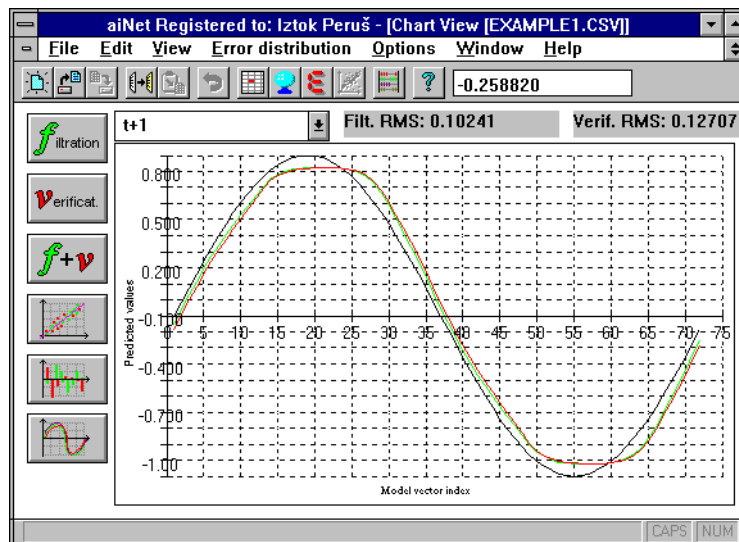


Figure 1.13

Note : Figure 1.13 shows example No. 1 from Part 3. Users can get the same results by using the value 0.03 for the penalty coefficient.

2.2 aiNet Menu

aiNet has been designed as a user friendly Windows application. The user commits all actions through the menu commands or by pressing speed bar buttons. aiNet uses two kinds of menus: static

and dynamic. Static menus are always present in the menu bar and are almost always accessible. Dynamic menus are view-dependent. When the *Model Vector View* window is active, for example, the *Edit* menu and *Model Vectors* menu appear on the menu bar, but they disappear when some other views are active.

2.2.1 Static Menus

Here is a table of the static menus and associated commands.

File	View	Options	Window	Help
New	Model Vectors	Comment width	Cascade	Contents
Open	Prediction	Fonts	Tile	About
Close View	Error Distribution	Register	Arrange icons	
Save	Charts		Close All	
Save As				
Save Prediction				
Open Prediction				
Exit				

File Menu (see Figure 1.14)

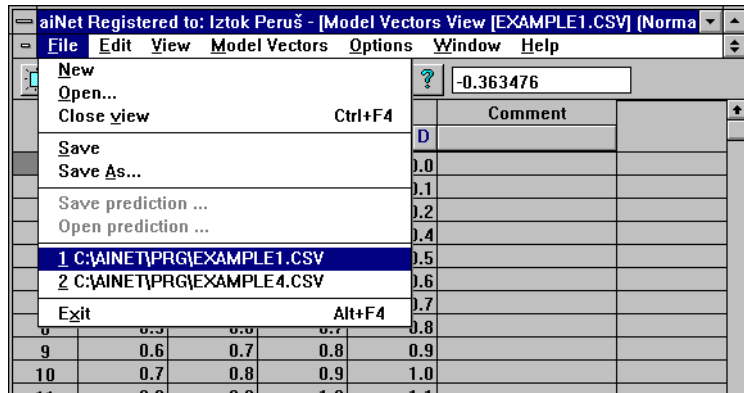


Figure 1.14

New

Using this command aiNet opens a new neural network. A dialog asks for the number of model vectors and the number of variables in the model vector. (These may be altered later.)

Once the number of vectors has been defined, aiNet will ask for the file format required. Select the required format and then the *Model Vectors View* window will appear on the screen. Data may now be entered into aiNet.

Open

Use this command to load an existing neural network from disk. aiNet will display the standard *File Open* dialog, enabling the file to be selected. If the file is loaded successfully, the aiNet will automatically open the *Model View* and display model vectors from the file.

Close Neural Net

Close Neural Net command will close all opened views of the current neural network and delete the network from memory. A confirmation of this operation is required.

Close View

aiNet will close the active view of the current neural network. If this view is the last opened view, a prompt is given to save the neural network to the disk.

Save

Save command will save the current neural network to the disk. If the neural network does not have a file name yet and is labeled as Untitled1 or Untitled2 ... then *Save As* command will be executed.

Save As

This command saves the current neural network to the disk, allowing the user to enter a filename and location. The file format, either the binary format (AIN extension), or the ASCII comma delimited format (CSV extension) may be selected. To use the CSV format, the neural network must be denormalized first.

Save As command also changes the text in the title bar of the views -- a new file name will appear within the square brackets.

Save Prediction

This command is enabled only if the *Prediction View* is active. *Save Prediction* allows the prediction vectors to be saved separately from the model vectors. This is useful if several different independent sets of prediction vectors are required. Predictions are always saved in the ASCII comma delimited format, with a .PRD extension.

Open Prediction

This is only enabled if the *Prediction View* is active. The current *Prediction View* will be overwritten with the prediction vectors from the file.

Exit

Use *Exit* when to quit aiNet. Before aiNet closes, it will ask for confirmation. A prompt to save any new or modified files is also displayed.

View Menu (see Figure 1.15)

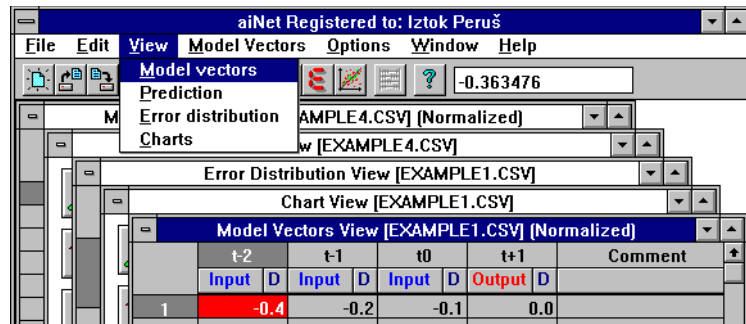


Figure 1.15

This menu allows a view to be selected. If the view has not been created, then aiNet creates a window for the selected view and puts it on the top. If the view window exists, then this command puts selected view window to the top.

Sometimes some of the views may be disabled. This usually happens when the model vectors in the *Model Vectors View* window are not normalized. *Selecting Model Vectors|Normalize + Lock* command will enable all views.

Options Menu (see Figure 1.16)

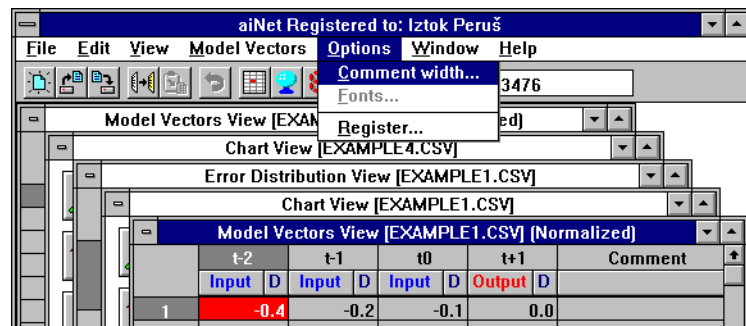


Figure 1.16

Comment Width

This allows the number of characters in the comment field to be defined.

Each model vector may have a comment. The model comments may be used to cross reference to experimental data, such as time and date, thus allowing efficient correlation.

Fonts

In aiNet versions 1.2x this command is disabled.

Register

When aiNet is registered, a code is provided which will prevent the registration dialogues from appearing. Enter your name and code in a dialog box, which is invoked by *Register* command. See : All About Registration, page 43.

Window Menu (see Figure 1.17)

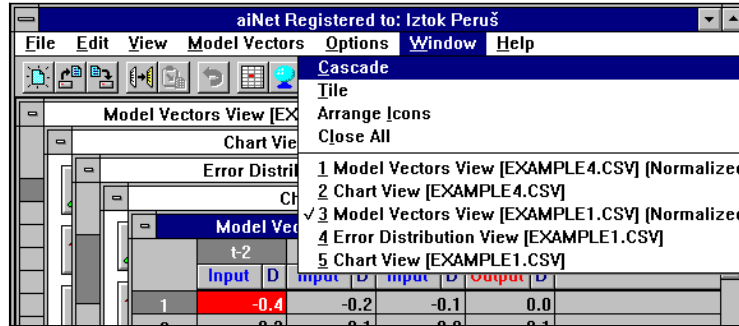


Figure 1.17

This is a standard window menu, whose commands are used to arrange and display child windows -- views over the application area. Commands *Tile* and *Cascade* rearrange the positions of the views. *Arrange Icons* does the same for the icons.

Command *Close All* closes all the views. A save prompt for opened neural nets is displayed.

Help Menu (see Figure 1.18)

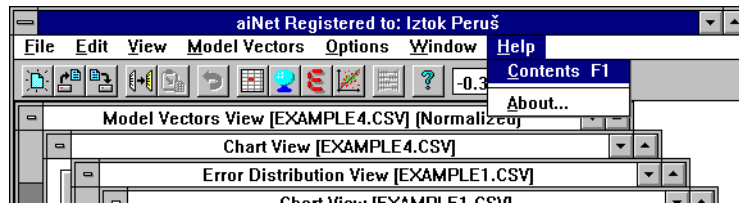


Figure 1.18

This is a typical Windows Help menu.

2.2.2 Dynamic Menus

The dynamic menus are defined below. The dynamic menus are activated by particular views and are not always displayed.

Edit Menu & Model Vectors Menu (see Figure 1.19)

Edit	Model Vectors
Add rows (model vectors)	Normalize + Lock
Add columns (variables)	Denormalize + Unlock
Insert row/column	Normalization settings
Delete row/column	View statistic...
Delete empty rows (model vectors)	Include selected rows
Copy	Exclude selected rows
Paste	Delete all excluded rows
Select all	Include all rows

Add rows (model vectors)

aiNet allows additional rows (Model vectors) to be added.

aiNet will open a simple dialog. The dialog will ask for the number of rows (model vectors) to be added. Enter the number and close the dialog with the *OK* button. The selected number of new rows will appear at the end of the *Model Vector View*.

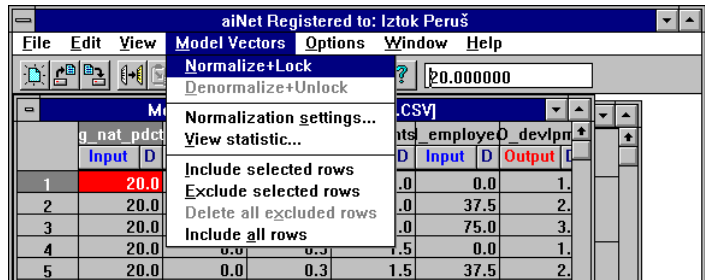
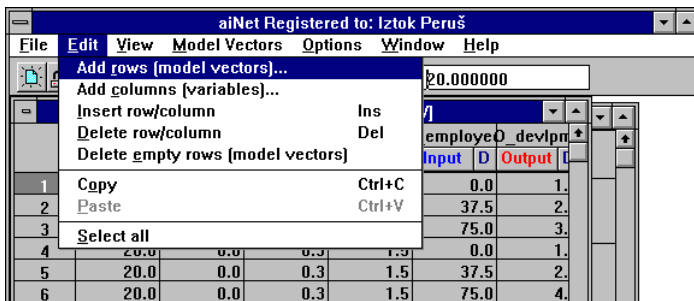


Figure 1.19

Add columns (variables)

aiNet will allow additional variables to be added to the model vectors. A dialog will pop up asking for the number of new variables. All of the new variables will have the 'E' exclude status turned on. (Exclude means neither input or output) and this should be defined as required.

Insert row/column

This command is only available if a model vector (row) or a variable (column) has been preselected. If a row has been selected, then a new row (model vector) will be inserted before the selected row.

The same is valid for the columns; a new column (variable) will be inserted before the selected column. The new variable will have the exclude status turned on.

Delete row/column

This command is only available if a model vector (row) or a variable (column) has been preselected. Before the selection is deleted, aiNet will ask for confirmation.

Delete empty rows (model vectors)

This command deletes all model vectors with no entries. It is good practice to delete empty rows as this will not only free memory, but it will also speed up computation.

Copy , Paste

Copy and *Paste* work in a similar manner as in most spreadsheet applications. Both commands may be also selected by the corresponding icons. *Copy* and *Paste* are mainly used for data transfer to and from different spreadsheet applications

NOTE: *Copy* and *Paste* commands may also be used within aiNet. At least one row (model vectors) must be available for the *Paste* to work. Use the command *Add rows (model vectors)...* in the *Edit* menu if necessary. aiNet will then automatically adjust the number of rows and then paste in the selected model vectors.

Normalize + Lock

aiNet requires the vectors to be normalized before computing. After the normalization, aiNet will enable calculation and prediction commands. This command also locks the model vectors - they cannot be edited until they are denormalized.

Denormalize + Unlock

This command reverses normalized model vectors back to their original values. It also unlocks them and thus allows them to be edited. With unlocked vectors, aiNet is unable to perform calculations.

Normalization settings

Since this command has a major influence on the results and since all views enable this command, it is fully explained later in the text. See *Three Most Important Commands*, page 21.

View statistic

This command shows simple statistical information. (see Figure 1.20). The information window shows minimum, maximum, mean, standard deviation and the missing value of each selected variable.

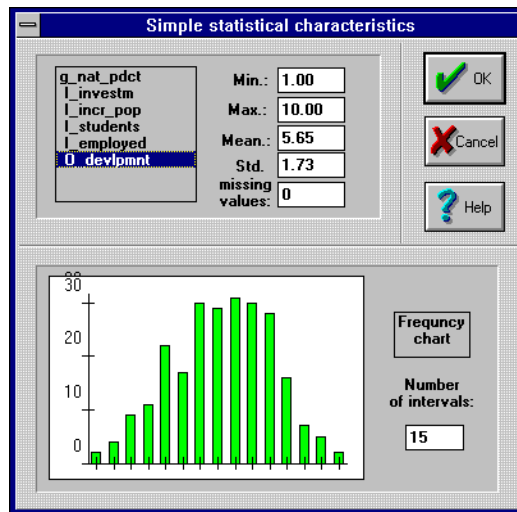


Figure 1.20

Graphic presentation of statistical information shows corresponding Frequency Chart in the same window. The user may choose the number of intervals, which should be between 3 and 50.

Include selected rows

This command will include selected rows (model vectors) back to the model. It has influence only if selected model vectors were previously excluded.

Exclude selected rows

This command will exclude selected rows (model vectors) from the model. It has influence only if selected model vectors were previously excluded. Model vectors that are excluded are not taken into account when prediction or error distribution is calculated.

Delete all excluded rows

All model vectors that has been marked as excluded will be permanently removed from the model.

Include all rows

All model vectors will be included into the model by deleting exclude flag from all previously excluded model vectors.

Edit Menu & Prediction Menu (see Figure 1.21)

Edit	Prediction
Show local error	Calculate prediction
Hide local error	Penalty settings
Add rows (pred. vectors)	Normalization settings
Insert rows (pred. vectors)	Local error settings
Delete rows (pred. vectors)	
Delete empty rows (predictions)	
Copy	
Paste	
Select all	

Show local error

aiNet can besides prediction, also calculate an estimation for the error (or the noise rate) for every single prediction result. The *Show local error* command displays these estimations in the *Prediction View* window. In order to see these estimations, the error distribution must be calculated first. If the error distribution is not calculated, then only empty lines will be displayed.

Hide local error

This performs the opposite of the *Show local error* command. It hides error estimations in the *Prediction View* window.

Add rows (pred. vectors)

This command is the same as the *Add rows (model vectors)* command, except that it acts in the *Prediction View* window.

Insert row (pred. vectors)

A row must be selected, before this command may be used. It will insert a new prediction vector before the selected row.

Delete row (pred. vector)

A row must be selected, before this command may be used. It will delete the selected row. Confirmation of this command is required.

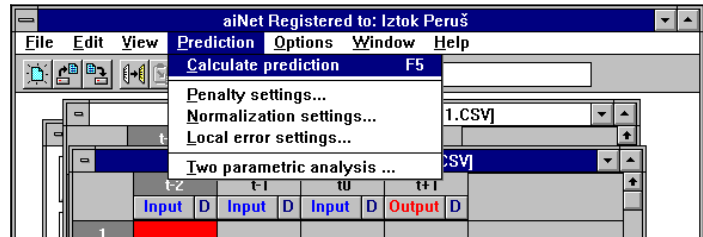
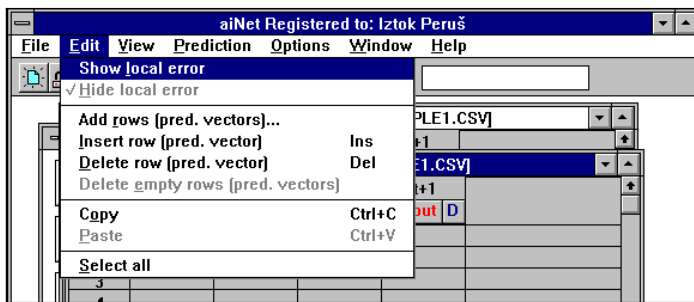


Figure 1.21

Delete empty rows (pred. vectors)

This command is the same as *Delete empty rows (model vectors)* command, except that it acts in the *Prediction View* window.

Copy , **Paste** 

Copy and *Paste* work similar as they work in most spreadsheet applications. Both commands may also be selected by the corresponding icons. *Copy* and *Paste* are mainly used for data transfer to and from different spreadsheet applications.

Calculate prediction  (shortcut - F5)

When all of the values have been entered into the prediction vectors, the output values may be calculated by selecting the *Calculate prediction* command. The results appear in the output variables of the *Prediction View* window. There may be a short delay whilst the calculations are performed. If the error distribution was previously calculated, then the estimation of the errors will also be calculated. The error distribution will be shown, if *Edit|Show local error* is selected.

Penalty settings

The Penalty setting is one of the most important commands of aiNet. It provides a dialog box, where the penalty coefficients and flags may be set and changed. This is fully defined later in the text. See *Three Most Important Commands*, page 21.

Normalization settings

Since this command has a major influence on the results and since all views enable this command, it is fully defined later in the text. See *Three Most Important Commands*, page 21.

Local error settings

This command has major influence on the results and is explained later in the text. See Three Most Important Commands, page 21.

Edit Menu & Error Distribution Menu (see Figure 1.22)

Edit	Error Distribution
Show difference	Calculate error distribution
Show predicted value	Global error report
Copy	Penalty settings
	Normalization settings
	Local Error Settings

Show difference & Show predicted value

As previously described, aiNet can calculate an estimation for the error in the prediction. Before it can do that, the error distribution (filtration and verification) must be computed. *Error Distribution View* is used to show the results of filtration (FE:) and verification (VE:). These results may be presented in two ways:

- As the difference between an initial (correct) and a calculated result (*Show difference* command),
- as a predicted value - calculated result (*Show predicted value* command).

NOTE: Verification and filtration are two special kinds of prediction. See User's Manual, Part 2: Basics About Modeling with aiNet, Chapter 3.

Copy

See The Copy Command in Different Views, page 34.

Calculate error distribution

This command engages the calculation of error distribution. The results are displayed in the output variables of the *Error Distribution View* window. If there are many model vectors, the calculation of the error distribution may take a considerable amount of time. (**NOTE:** Doubling the number of model vectors, causes the calculation to take four times as long.)

Global error report

This command provides a global error estimate (root mean square error = RMS). The result is displayed in a dialog, where the total RMS error^{##} is shown, together with the RMS errors for single output variables. See also Part 2: Basics About Modeling with the aiNet, Chapter 3.

^{##} 'total RMS' means one error for all output variables

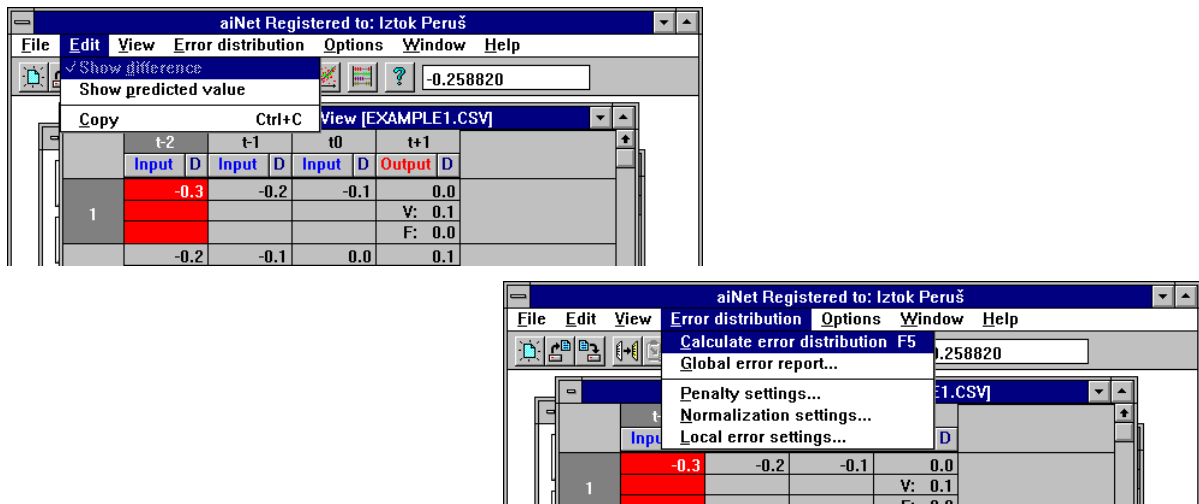


Figure 1.22

Penalty settings

Penalty settings are one of the most important commands of aiNet. It provides a dialog box, where the coefficients and flags may be set and changed. This is fully explained later in the text. See Three Most Important Commands, page 21.

Normalization settings

Since this command has major influence on the results and since all views enable this command, it is fully defined later in the text. See Three Most Important Commands, page 21.

Local error settings

This command has major influence on the results and is fully explained later in the text. See Three Most Important Commands, page 21.

2.2.3 Three Most Important Commands

One of the features of aiNet is that it uses only one coefficient, which has major influence on the results. Besides this coefficient, there are also a few commands, which effect the results. These commands are now defined in more detail.

Normalization Settings

When this command is selected, it provides a dialogue. It allows the choice of one of two kinds of normalization procedures:

- regular
- statistical

Before aiNet can perform any calculations, all of the model vectors must be normalized. This process transforms real values into values that aiNet can process. Two normalization methods are available as defined below.

With regular normalization, aiNet will perform the following actions.

1. Scan through all model vectors and find the largest (maximum) and the smallest value (minimum) for each variable.
2. For each variable, calculate normalization factors in such a way, that all values in model vectors will be between -1 and 1 after normalization. (The maximum will be 1 and the minimum will be -1.)
3. Normalize all model vectors. Now all values in model vectors will be between -1 and 1.

Statistical type of normalization is different. aiNet will perform the following during statistical normalization:

1. Run through all model vectors and calculate an average and a standard deviation for each variable respectively.
2. For each variable calculate normalization factors in such a way that values, which are exactly one standard deviation apart from the mean value, will become 1 (or -1) after normalization.
3. Normalize all model vectors. All values that are now less than $|1|$ are those within one standard deviation, all others (greater than $|1|$) are more than one standard deviation apart from the mean value.

The following generalization may be applied when choosing the most appropriate rule:

Use regular normalization if the data in model vectors is already normalized in some way -- if all values fall within some physical limits.

When the values do not necessarily fall within limits, use statistical normalization.

In most cases the normalization type does not reflect significantly on the results. Note: normalization type does have an influence on the optimal value of penalty coefficient. Statistical normalization usually requires larger values.

Penalty Settings

This is the most important command. When selected this command provides a dialog. The dialog asks for the value of the penalty coefficient to be entered and also lets the type of the coefficient be selected.

The Introduction describes that there is an optimal value for the penalty coefficient. The penalty efficient curve is, however, shallow and therefore valid results may be obtained for a wide range of coefficient values.

The following guidelines may be used to assist with determination of the coefficient:

- The more model vectors, the smaller the optimal value.
- The more variables, the larger the optimal value.
- The more noise in the data, the larger the optimal value.
- Experiment with different values, including extreme values.
- Small penalty value lead to overfitting (overtraining).
- Large penalty value lead to underfitting (overgeneralizing).

The dialog which pops up gives three possible choices for the penalty coefficient type (see Figure 1.23):

- static
- dynamic

- nearest neighbor

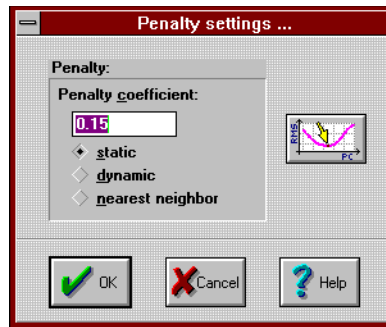


Figure 1.23

Choosing coefficient types.

The static coefficient type defines a single penalty coefficient value for all of the hyper-dimensional space. This type of coefficient should be used when the model vectors are approximately uniformly (regularly) distributed. Use this selection for noisy data.

The dynamic coefficient behaves differently. When prediction, filtration and verification are calculated, then actual value of penalty coefficient depends on the density of model vectors. If there are a lot of model vectors in the surroundings of the prediction vector, then the penalty coefficient decreases, to fit more. However, if model vectors are rare in the surroundings of prediction vector, then the penalty coefficient increases and generalizes more. Use the dynamic coefficient type, when the model vectors are grouped in clusters and are not uniformly distributed over hyper-dimensional space.

The nearest neighbor type of penalty coefficient may produce a more efficient model in certain cases. The penalty coefficient is determined as a function of the distance to the nearest model vector in the model hyperspace. More details will be given in the next release of the aiNet. Again, experiment with this setting to determine it's suitability for the application.

By changing the penalty coefficient type from dynamic to static will cause the optimal value of the coefficient to change. The dynamic type usually leads to smaller values.

aiNet also has an automatic method of determining the optimal value. Selecting the button on the right side (see Figure 1.23), will cause aiNet to find an estimation of the optimal coefficient value. The automatic determination of the penalty coefficient is based on minimizing a RMS verification error.

The sub-window for the penalty coefficient optimization is shown in Figure 1.24. Default values are already set for the search range and precision (lower & upper search bounds, precision). These may be altered to suit the particular problem.

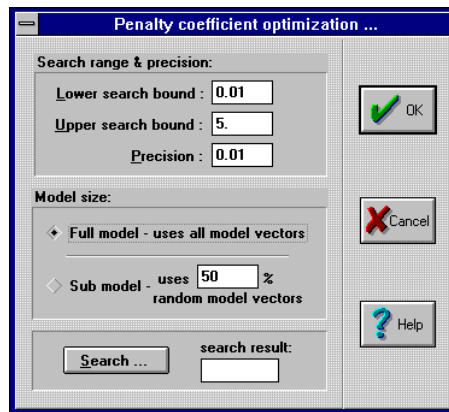


Figure 1.24

NOTE: The model size setting can be very important. Full model uses all model vectors, while sub-model uses just a given per cent of randomly chosen model vectors (default value is 50 %). The first option may be used in all cases, where the data base is relatively small (small number of model vectors and/or small number of input/output variables). In cases of large data bases, the second option is highly recommended. This option may be computational intensive and time consuming.

Error Distribution Settings

The error distribution and a local error (or local error estimation), are related and have the same background.

Error distribution is used together with the terms filtration and verification. The error distribution is the result of filtration and verification. It is named distribution, because the filtration and verification calculate an error for each model vector. When the error for each model vector is known, it's distribution over all of the model vectors is also known, hence the term distribution.

Local error estimation is associated with prediction. If the error distribution is known, and the prediction process is run, then not only are the output variables predicted, but also the error for each prediction. Because such an error prediction can behave locally (it is not a constant in hyperspace), the term local error is used. Since error distribution is always noisy, this error prediction cannot be very accurate, hence the term estimation.

Using the *Error Distribution Settings* command, the calculation of the error distribution may be controlled. The error distribution may be set as follows:

- *Enable (Yes, calculate local error),*
- *Disable (Do not calculate local error).*

The type of local error may also be as follows:

- *Unsigned (absolute) local error,*
- *Signed local error.*

Use an unsigned (absolute) type for noisy model vectors. When the problem is smooth and without any noise, then select the signed local error type.

The selection of local error type will have the greatest effect on the prediction process, more precisely on the estimation of local error in the prediction process.

2.3 A more complex Problem

The previous section dealt with aiNet's commands. These will now be used to solve a more complex problem.

2.3.1 The Hole in a Square Problem

Imagine a two dimensional square region, which has a hole in the middle. The square region is placed into a Cartesian coordinate system, so that every point in the region can be located knowing the point's coordinates. The example shown in Figure 1.25, shows a two unit square and a hole, 1.4 units in diameter.

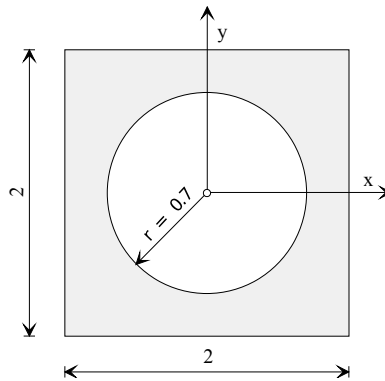


Figure 1.25: The hole in a square problem.

Select one random point in the square region. Given the coordinates of the point, determining whether this point is in the hole may be accomplished by substituting the values into the equation:

$$r = \sqrt{x^2 + y^2}$$

if $r \leq 0.7 \rightarrow$ point (x, y) lies inside the hole

if $r > 0.7 \rightarrow$ point (x, y) lies outside the hole

aiNet may be trained to do the same job.

aiNet must therefore be provided with some training examples -- model vectors. Since there many combinations of model vectors, these will be generated rather than entered directly.

2.3.2 Generating Model Vectors

A spreadsheet will be used to generate the vectors. A random generator function is used to provide 200 random points in the square region. The spreadsheet will also be used to determine whether the point is inside or outside the circle. Points lying inside the hole are assigned a value 1, points lying outside the hole, are assigned a value 0. Each model vector has three variables: x coordinate, y coordinate and a flag indicating the point's position.

Table 1.3: Hole in a square problem - input and output variables.

x (input)	y (input)	where is it (output)
random x	random y	1 or 0
...

random x	random y	1 or 0
----------	----------	--------

Used random points are presented in the Figure 1.26.

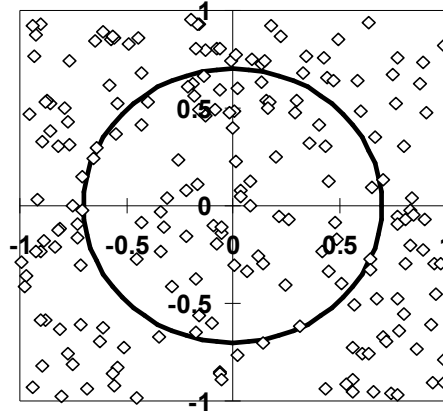


Figure 1.26: The hole in a square problem - database.

These model vectors are provided in the example file HOLE.CSV, provided with aiNet. Open this file by selecting the *File|Open* command and selecting HOLE.CSV. Close the dialog with the *OK* button.

This provides the model vectors for aiNet. Modeling will now be performed on the data.

2.3.3 Modeling with aiNet

Modeling with aiNet is a simple iterative procedure. First the model vectors are normalized, then an optimal penalty coefficient is iteratively selected and finally the predictions are calculated.

Normalization

Since the coordinates are uniformly distributed, regular normalization may be used. Select the *Model Vectors|Normalize* settings command and turn the option named *regular* on.

To normalize model vectors, select *Model Vectors|Normalize* command and then switch to the *Charts View* window.

Optimal Penalty Coefficient

Select the *Error Distribution|Penalty settings* command to display the dialog. Before choosing values for the penalty coefficient, select the penalty type. Since there are a few clusters of points and also a few white areas in the Figure 1.26, choose the dynamic penalty type. (Note, in this example this is not very important.) When the penalty type is set, select a value for the penalty coefficient.

The best way to obtain the optimal value for penalty coefficient is to run filtration and verification and then to observe the RMS verification error. The value which gives the smallest RMS verification error will be usually very close to the optimal penalty coefficient.

Table 1.4: The hole in a square problem - test results.

Penalty coefficient	Verification RMS error
0.04	0.3585
0.06	0.3442
0.07	0.3350
0.08	0.3303
0.09	0.3323
0.10	0.3400
0.15	0.4137
0.30	0.6181

In order to find the value which will minimize the verification RMS error, select several values for the penalty coefficient and run filtration and verification for each value. To do this, enter a value in the *Penalty Settings* dialog and then select *the Error Distribution|Calculate Error Distribution* command if necessary. The results are presented in the Table 1.4 and Figure 1.27.

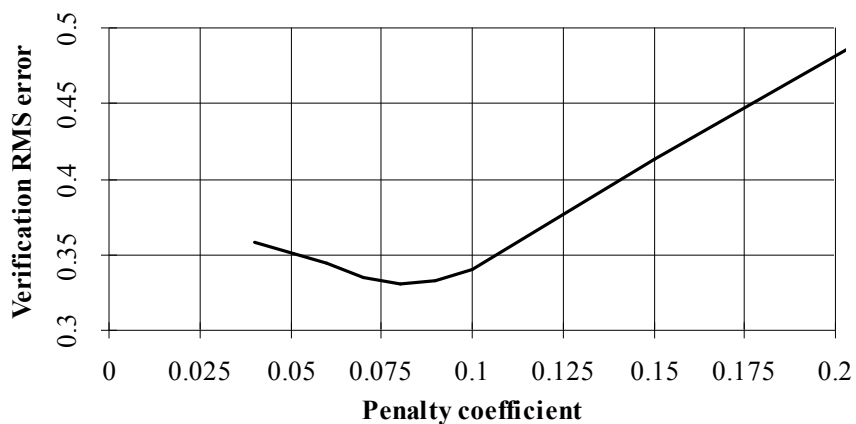


Figure 1.27: Penalty coefficient via verification RMS error.

A good estimation for the penalty coefficient would be 0.08 according to the graph shown in Figure 1.27. Note that the optimality curve is very shallow at the minimum.

NOTE: The automatic method of penalty coefficient estimation may also be used with similar effect.

Verification of the Model

Once the optimal value of penalty coefficient is found, the modeling is complete. aiNet may now be tested with new data to verify its performance.

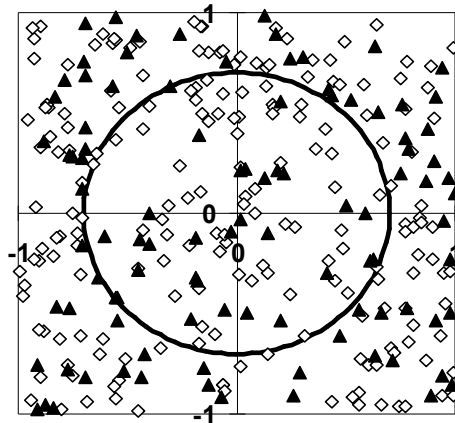


Figure 1.28: The hole in a square problem - test example.

To do so, a 100 new random points will be generated, which will be distributed over our square region. For these points, exact results will be calculated and aiNet will also make predictions. The results will then be compared.

100 new prediction vectors were generated using the spreadsheet and these are provided in the HOLE.PRD file. The new points (prediction vectors) are shown together with model vectors in the Figure 1.28.

To calculate predictions for new vectors, bring the *Prediction View* window to the front by selecting the *View|Prediction* command. Load the HOLE.PRD file into the *Prediction View* by selecting the *File|Open Prediction* command. In the dialog, select the HOLE.PRD file and close the dialog with the *OK* button.

Check if the penalty settings are correct by selecting the *Prediction|Penalty Settings* command and ensuring that the dynamic penalty type is set to 0.08. Close the dialog and select the *Prediction|Calculate Prediction* command.

aiNet will calculate the predictions and will display them in the *Prediction View* window.

Remember, points inside the hole are assigned a value 1 and points outside the hole are assigned a value 0. aiNet cannot give an answer as sharp as the equation can provide. Instead of it will give a value between 0 and 1. Values below 0.5 represent points outside the hole and values above or equal 0.5 represent points inside the hole. To see the values represented by either 0, or 1, double-click the output variable name "Result" and select 0 in the *Number of decimals* edit box.

Knowing the exact results from the equations, aiNet's performance may now be judged. Comparing the results, it can be seen that aiNet failed to predict three from 100 points. This means a 97% success⁺⁺. Furthermore, switching to the *Error Distribution View* and counting how many errors were made in the verification boxes (VE: boxes), we find 7 such model vectors and this means a 96.5% success. This shows that the penalty coefficient value determined on the basis of verification and filtration is a good estimation.

⁺⁺ One hundred predictions vectors is a small test set. To be more sure of aiNet predictions a larger set should be used.

The prediction test was repeated with different values for the penalty coefficient and the results are shown in the table below.

Table 1.5: The hole in a square problem - test results via penalty coefficient.

Penalty coefficient	Failed predictions
0.05	2 (2%)
0.08	3 (3%)
0.15	6 (6%)
0.30	7 (7%)

2.3.4 A Hole in a Square Problem with Noisy Data

The effects of noise upon the model vector data will now be examined by modifying the previous example. Noise will be added to the third variable “Result”. The noise will be added to all of the 200 model vectors, simulated with gaussian distribution, where the mean value is 0 and standard deviation is 0.5. This gives a significant noise factor, considering that the original results had values of either 0 or 1.

The Model vectors for this example are provided in the HOLE-NS.CSV file.

Optimal Penalty Coefficient

Using the same procedure as in the previous example, the optimal value for the penalty coefficient is found to be 0.16.

Verification of the Model

For verification, the same prediction vectors of the previous example are used. Specifying 0.16 for the penalty coefficient and calculating the prediction, provides good results. aiNet fails in only four cases, which means a 96% success.

The prediction test was repeated with different values for the penalty coefficient. The results are presented in table 1.6 below.

Table 1.6: The hole in a square problem (noisy data) - test results via penalty coefficient.

Penalty coefficient	Failed predictions
0.05	11 (11%)
0.10	7 (7%)
0.16	4 (6%)
0.20	5(5%)
0.25	5 (5%)

2.3.5 Some Possible Improvements

There are ways to improve aiNet’s performance. The following are two examples.

The best way of improving the prediction performance is adding new model vectors to the data. In reality, this is done by collecting additional information about the problem. The data is then transferred into new model vectors or even into some new variables in the existing vectors.

If unlimited model vectors are available and the problem is smooth, then the results may be improved using the following procedure.

1. Put a small number of model vectors into the neural network knowledge base.
2. Use some new model vectors, which are not in the data base and use them as prediction vectors.
3. Run a prediction and compare the predicted result (prediction vectors) with the correct results (model vectors).
4. Add those model vectors with the biggest difference between predicted and correct results into the data base.
5. Repeat the steps 2, 3 and 4 until the desired precision is obtained.

This method of improvement works well in smooth examples, but it is untested with noisy examples.

2.3.6 Final Comment

It is important to emphasize one very important thing about neural networks. Even with a high performance system such as aiNet, incorrect results will be obtained if the data (model vectors) are unsuitable. This means that in order to obtain good results, the model vectors should be chosen carefully. Are there too many variables in a model vector or too few? Are the model vectors complete? Is the source of the data reliable? With reliable vectors, the aiNet process is simplified and robust.

aiNet, like any other neural network application, is just a tool. It's major feature is simplicity and its orientation toward modeling, allowing the user to concentrate on the model vectors and data.

2.4 How Can I Use My Data with aiNet ?

Many users have their data already organized in an ASCII form and it would be convenient to use this data in with aiNet. A spreadsheet application may be used as a converter. *Microsoft Excel* for example, may be used for this purpose. Excel has a wizard which takes user data and converts it into individual cells. Once the data is properly converted and loaded into the spreadsheet, the *Copy* and *Paste* commands may be used to transfer the data into aiNet.

An example of this procedure using a file named as MULTI.DAT is shown on in Figure 1.29.

1	1	1
2	1	2
3	1	3
4	1	4
1	2	2
2	2	4
3	2	6
4	2	8
1	3	3
2	3	6
3	3	9

4	3	12
1	4	4
2	4	8
3	4	12
4	4	16

Figure 1.29: File MULTI.DAT in ASCII form. This file will be imported into aiNet.

This file holds multiplication combinations of integers from 1 to 4 and the result of the multiplication. This file may not be directly loaded into aiNet, but it may be loaded via *Microsoft Excel*. Start *Excel*. The screen will be similar to that shown in Figure 1.30.

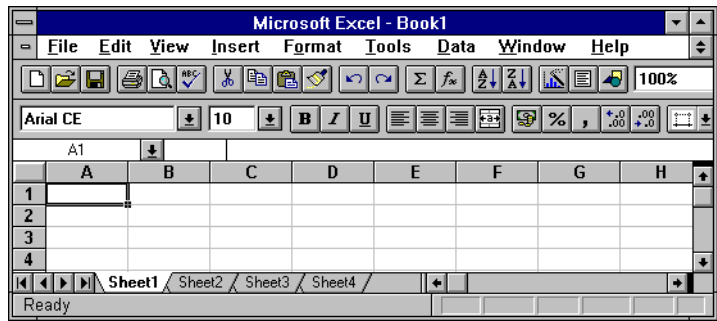


Figure 1.30: Microsoft Excel.

Now select the *File|Open* command and open the MULTI.DAT file, which is in the AINET\EXAMPLES directory. (Select *All Files (*.*)* in the *List Files of Type* combo box to see all files in the directory.) A Text Import Wizard appears on the screen as shown in Figure 1.31.

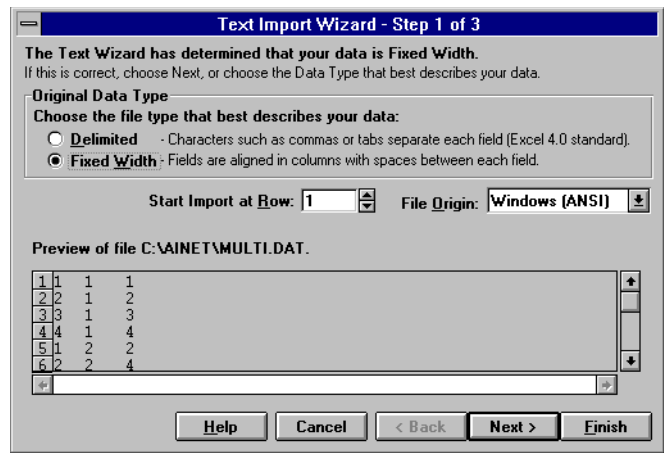


Figure 1.31: Excel Text Import Wizard

Generally, follow the instructions provided by the application. In our example, press the *Next* button. Second step of the wizard appears on the screen, press the *Next* button again. Now press the *Finish* button to import the data into Excel. The screen should now appear as shown in the Figure 1.32.

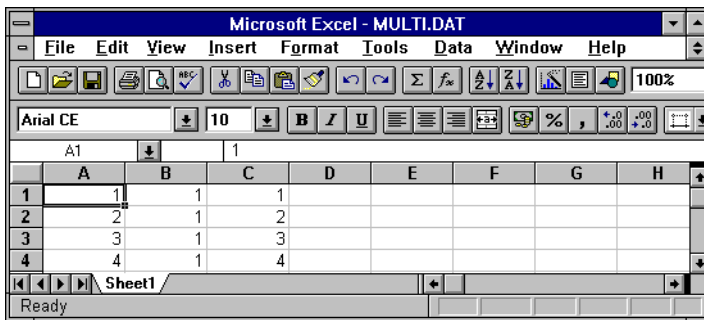


Figure 1.32: Excel with imported MULTI.DAT file

Now that the data is located in cells, it may be transferred to aiNet. Select all of the cells for transfer to aiNet and execute the *Edit|Copy* command. Start aiNet and select the *File|New* command. aiNet asks for the number of model vectors and variables. Simply enter nominal values, there is no need to be accurate with the entries.

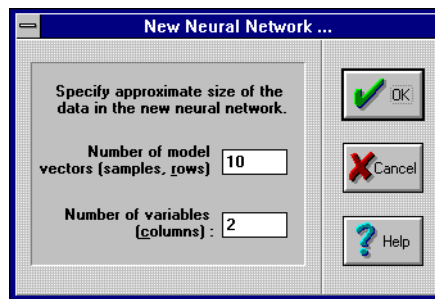


Figure 1.33: Approximate number of rows and columns.
It is better to specify smaller numbers.

aiNet will also ask for the file format. This selection is not important and any option may be selected. The screen should look like Figure 1.34.

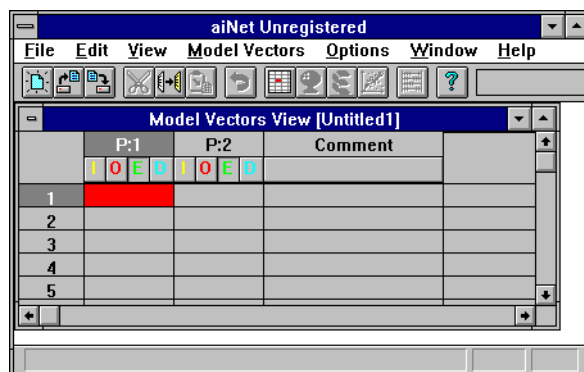


Figure 1.34: Empty model in the aiNet Model Vectors View

Select the top, left hand cell (as shown on Figure 1.34) and paste the data from the Clipboard using the *Edit|Paste* command. The *Paste* command also works in the comment section, which means that last column from the data in the Clipboard may be placed on comment section. Depending upon the source of the data, this may or may not be desired. In this example, the last column represents the

result of multiplication and it should not be interpreted as a comment. Therefore, answer **NO** to the overwrite question aiNet asks (Figure 1.35).

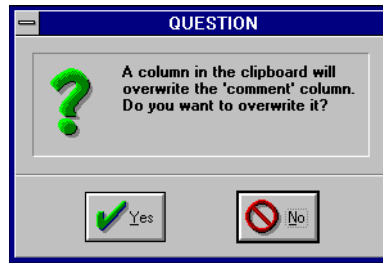


Figure 1.35: The overwrite Message box

aiNet determines whether there are enough cells in the model to paste data in. If there are not enough cells, aiNet provides a message box asking the user if necessary model vectors and/or necessary variables should be added. Examples of these message boxes are shown on Figure 1.36.

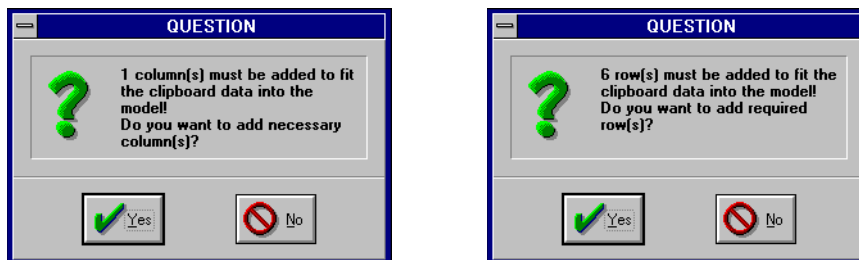


Figure 1.36: Message boxes

Answer Yes to allow aiNet to paste the new data. All of the selected data from the spreadsheet is now transferred to aiNet, which should now appear as Figure 1.37.

aiNet Unregistered - [Model Vectors View [Untitled1]]				
File Edit View Model Vectors Options Window Help				
1.0				
	P:1	P:2	P:3	Comment
	I O E D	I O E D	I O E D	
1	1.0	1.0	1.0	
2	2.0	1.0	2.0	
3	3.0	1.0	3.0	
4	4.0	1.0	4.0	
5	1.0	2.0	2.0	
6	2.0	2.0	4.0	
7	3.0	2.0	6.0	
8	4.0	2.0	8.0	
9	1.0	3.0	3.0	
10	2.0	3.0	6.0	
11	3.0	3.0	9.0	
12	4.0	3.0	12.0	
13	1.0	4.0	4.0	
14	2.0	4.0	8.0	
15	3.0	4.0	12.0	
16	4.0	4.0	16.0	

Figure 1.37: aiNet Model View after the data has been pasted

2.5 Printing and Copying

aiNet does not currently support direct printing, but instead relies upon the transfer of results to spreadsheet applications for further processing and printing. Copying the data into a spreadsheet is accomplished using the COPY command.

2.5.1 The Copy Command in Different Views

The actual data copied to the Clipboard depends on the currently active view. In the model Vector View and in the Prediction View, only the selected cells will be copied. In the Error Distribution View and in the Chart View, the data for whole charts are copied to the Clipboard.

2.5.1.1 Model Vector View and Prediction View

In the Model Vector View and in the Prediction View, only the selected cells are copied to the Clipboard. The cells can be selected by dragging the mouse from one corner to corner. All cells in dark rectangle are selected. The cells may also be selected by dragging the mouse over variable names (this selects whole columns) and over row numbers (this selects whole rows). See Figure 1.38.

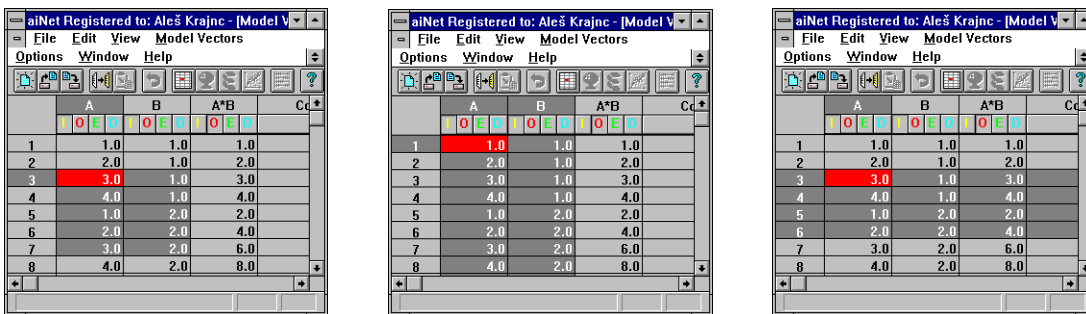


Figure 1.38: (a) a range of cells is selected, (b) columns are selected and (c) rows are selected.

Once the cells are selected, use the *Copy* command to copy the selection to the Clipboard. Switch to the spreadsheet, select a cell (top left) to paste the data and then use the *Paste* command.

2.5.1.2 Error Distribution View and Charts View

The use of the *Copy* command in these two views behaves differently - there is no need to make a pre-selection. The *Copy* command will copy the data in one of the three formats, which are defined by the three different chart views available in Charts View.

If the current view is Charts View, then the form of the copied data will correspond to the chart on the screen. Figure 1.39 shows how the data is transferred if the *Quasi Correlation Chart* is active in aiNet. Both the verification and filtration results were copied to the Clipboard because both were in the aiNet window. If *Verification* is selected before *Copy* has been selected, than only verification would be copied to the Clipboard. The data may be copied into Excel workbook to display the chart. To display a chart similar to the chart in aiNet, use the *XY chart* selection in the Chart Wizard.

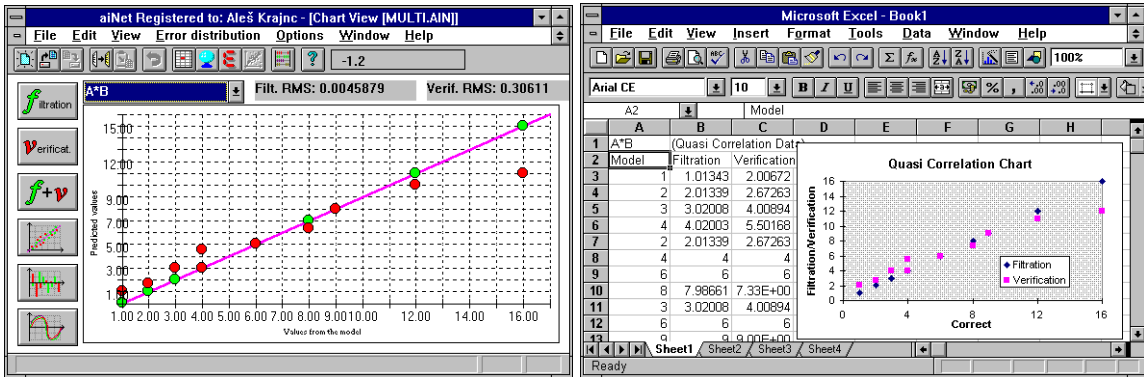


Figure 1.39: Quasi Correlation Chart copied from aiNet to Excel

If current aiNet chart is an *Error Chart* and if only the filtration results are shown in the aiNet window, then the *Copy* command will transfer the results in the form shown on Figure 1.40. To draw the chart in Excel, paste the data first and then select *Columns chart* in Chart Wizard.

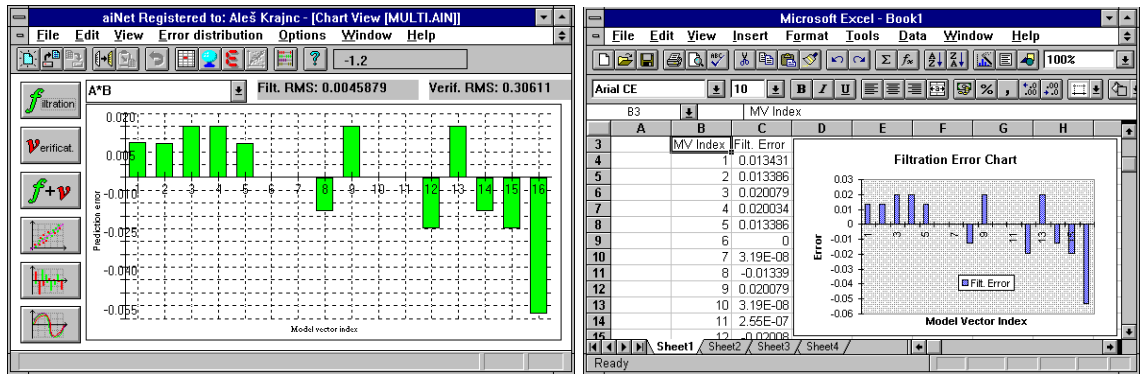


Figure 1.40: Error Chart copied from aiNet to Excel

Finally, if the *Line Chart* is current chart in aiNet and both the verification and filtration are active, then the *Copy* command will transfer results in the form shown on Figure 1.41. To draw the chart in Excel, the data must be pasted first and then *Line chart* selected in the Chart Wizard.

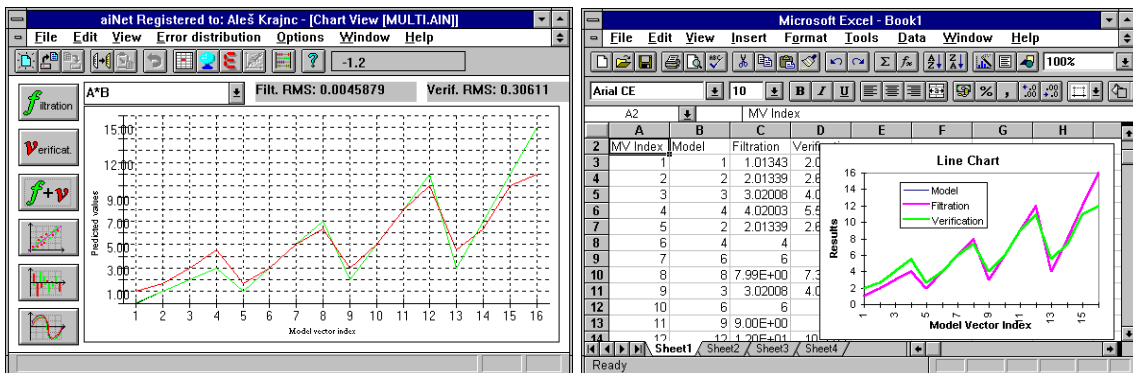


Figure 1.41: Line Chart copied from aiNet to Excel

If the *Copy* command is invoked with the Error Distribution View, then a dialog box pops-up (Figure 1.42) asking the user to define the source of the copy.

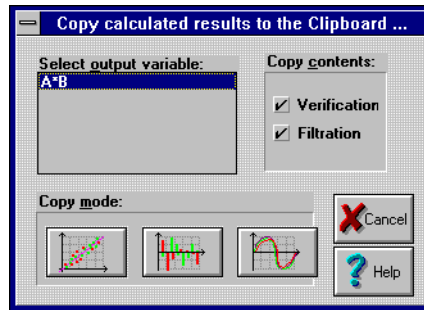


Figure 1.42: The *Copy* command invoked in Error Distribution View

2.5.2 Two Parametric Analysis - 3D Surface Chart

aiNet can also generate data for a 3D surface chart. To generate the data necessary for this chart, two inputs and at least one output variable must be selected. For each selected input variable, a range must be specified. aiNet will then calculate predictions of the output variables by changing the input variables over the specified range. Thus, prediction vectors will be generated and will be automatically loaded into the Prediction View. The Prediction View must be active before the *Two Parametric Analysis* command is used.

Once the prediction vectors are generated, the prediction is calculated and the results of the prediction of the selected output variables are copied to the Clipboard in a special format. These results must be pasted into a spreadsheet application in order to generate a surface chart.

The two parametric analysis takes two input variables (parameters) which values are changed within specified range and for each pair of values, output variable(s) are calculated. If there are more than two input variables, the other input variables need to be set to fixed values in order that they will act as constants. To do this, the input variables in the first row of Prediction View must be set to the desired values. This means that the values from the first row will be used as constants for input variables which are not used as parameters.

The *Two Parameters Analysis* command can be selected from the Prediction menu of Prediction View. If there are more than two input variables in the model, than the first set values of input variables in the Prediction View must be defined as constants. See Figure 1.43.

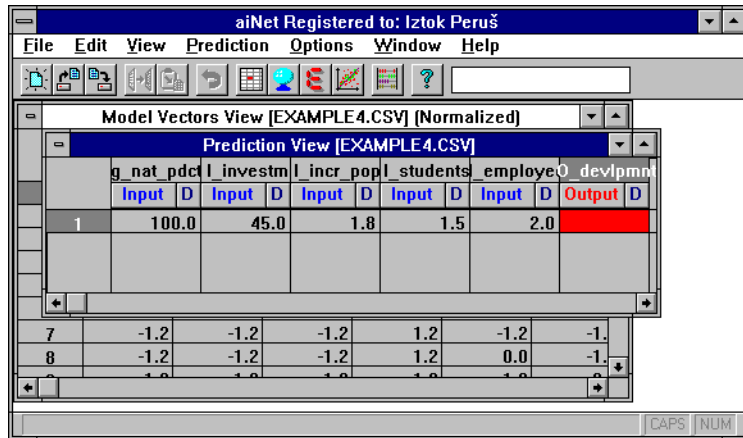


Figure 1.43: Setting constant values for input variables which will not be selected as parameters

By invoking the *Prediction|Two Parameters Analysis* command, a dialog box pops-up to enable the selection of input variables which will act as parameters. (See Figure 1.44). The range and number of intervals within each range must be specified, together with the Output variables. Once these are specified, the *Copy&Run* button will first generate prediction vectors needed for the calculation, then the calculation itself will be performed and the results will be copied to the Clipboard.

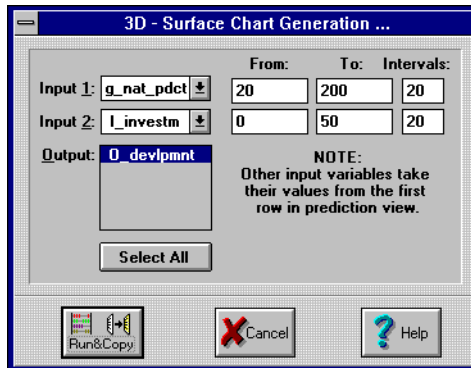


Figure 1.44: The Two Parameters Analysis dialog

Now Excel (or some other spreadsheet application) may be used to process the data generated. Select a cell and paste the results using the *Edit|Paste* command. Figure 1.45 shows an example of Excel with data ready to produce a surface chart.

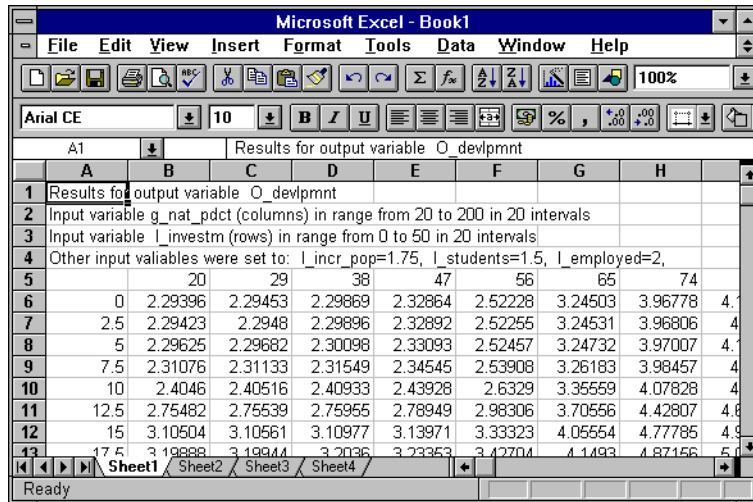


Figure 1.45: Excel and data pasted from the aiNet two parametric analysis.

In Excel, the numerical data must be selected and the Chart Wizard invoked. In the second step of the Wizard (Figure 1.46) the 3-D Surface chart must be selected. In the other steps, follow the instructions that the wizard gives. Finally a chart similar to one shown on Figure 1.47 is obtained.

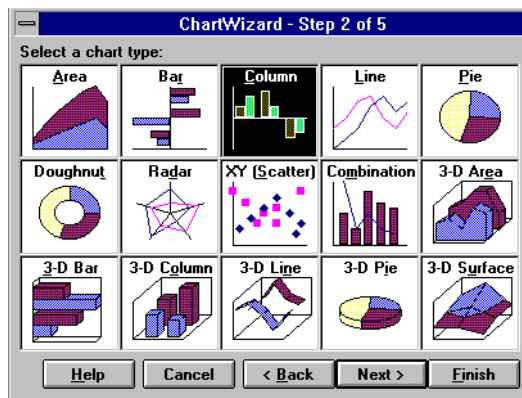


Figure 1.46: Selecting the 3-D Surface chart in Excel Chart Wizard

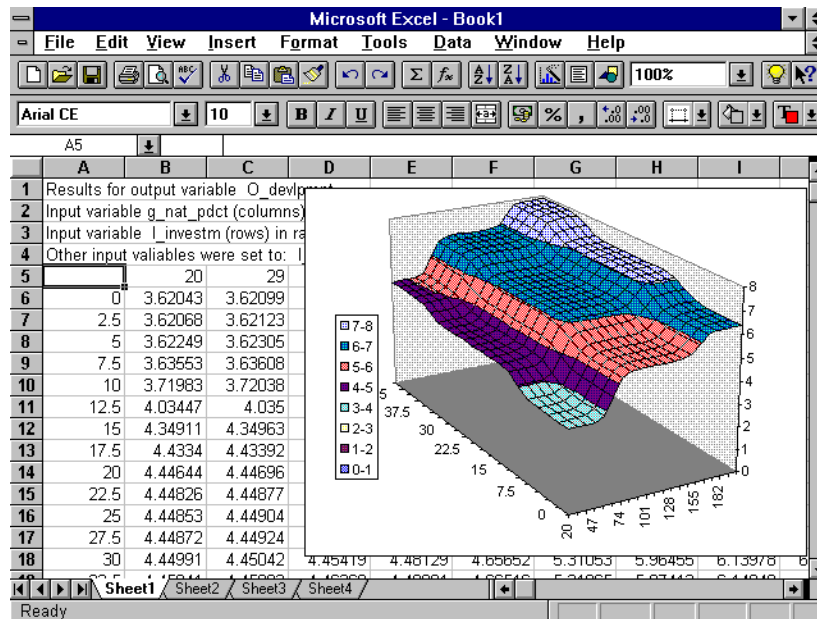


Figure 1.47: Final result - 3D Surface chart drawn from the data calculated in aiNet

2.6 aiNet's File Formats

aiNet's spreadsheet-like editor may be used to enter model vectors, or they may be imported in a suitable file format. aiNet supports two ASCII based formats: a CSV and a PRD format which are used for storing model vectors and prediction vectors, respectively.

2.6.1 The CSV File Format

A spreadsheet application is useful for data conversion as defined previously. The aiNet CSV (comma separated variable) format includes a header which should be added to the top of the data file.

Version 1.24 introduces a modified CSV file format. The first row holds the file version number. Additionally, flag has been added to each model vector. The first bit in the flag tells if model vector is excluded from the model (bit is set to 1) or not (bit is set to 0).

Old CSV format can be also used by any version of aiNet, however only version 124 or later will support the new format. Version 124 can not save the CSV file in old format.

The CSV file is defined as follows for the aiNet.

```

NEW row 1:    VERSION
      row 2:    P_COEF, RES, RES, NMV, NP, RES
      row 3:    PAR_NAME_1, PAR_NAME_2, ..., PAR_NAME_NP
      row 4:    PAR_ST_1, PAR_ST_2, ..., PAR_ST_NP
NEW row 5:    FLAG_1, MV_11, MV_12, ..., MV_1NP, Comment_1
      row 6:    FLAG_2, MV_21, MV_22, ..., MV_2NP, Comment_2
      rows :    ...
  
```

row 4+NMV: FLAG_NMV, MV_NMV1, ..., MV_NMV.NP, CommentNMV

Abbreviations :

NEW	VERSION	the CSV file format version
	P_COEF	penalty coefficient value,
	RES	reserved field. It should be set to 0 (zero).
	NMV	number of model vectors,
	NP	number of variables.
NEW	FLAG	a flag currently indicating only the status of model vector (excluded/included)
	PAR_NAME_j	the name of the <i>j</i> -th variable,
	PAR_STATUS	the status of the <i>j</i> -th variable,
	MV_ij	the value for the <i>j</i> -th variable in the <i>i</i> -th model vector,
	Comment_i	optional comment for the <i>i</i> -th model vector.

For the penalty coefficient, any value greater than zero may be used. The value may be changed later, from within aiNet. The type of penalty coefficient as well as the normalization type and the local error estimation type are set within aiNet, after reading the file.

All of the reserved fields should be set to zero.

The Variable status consists from two letters. The first letter indicates the status of the variable: I - input, O - output, E - excluded. The second letter indicates the type of variable: D - discrete, no letter or space - usual continuous.

The comment field is an optional part of each model vector. If there are missing values in the model vectors, then omit the value from the data. For example: 1, 2, 3, , 5 in the CSV file means 1,2,3, missing value, 5.

The file for the XOR example, which was presented in Chapter 1, is shown below.

124	124 is the CSV file version.
0.15, 0, 0, 4, 3, 0	0.15 - penalty coef., 4 - numb. of mv, 3 - numb. of param.
A, B, A XOR B	names of variables: "A", "B" "A XOR B", respectively
I, I, O	status of variables: input, input, output
0, 1, 1, 0	the first model vector (The first value is a flag!)
0, 1, 0, 1	the second model vector
0, 0, 1, 1	the third model vector
0, 0, 0, 0	the last model vector

Figure 1.48: Example of the CSV file and its description.

2.6.2 The PRD File Format

The PRD format is also a CSV format, but instead of the description of variables and model vectors it stores prediction vectors. The PRD format is as follows:

row 1: NPV, NP
row 2: PV_11, PV_12, ..., PV_1NP
row 3: PV_21, PV_22, ..., PV_2NP

```

ROWS : . . .
row 1+NPV:  PV_NMV1, PV_NMV2, . . . , PV_NMV.NP

```

Abbreviations:

```

NPV      number of prediction vectors,
NP       number of variables,
PV_ij   the value of j-th variable of the i-th prediction vector.

```

In the PRD file, the number of variables (NP) must be equal to the number of variables in the model vectors. This means that whole (input part+output part+excluded part) prediction vectors must be put into the file, and not just the input part. Any values may be assigned to the output values, because these values are overwritten during the prediction process.

An example of a PRD file, based on the XOR problem, is shown in Figure 1.49 below.

5, 3	5 - number of prediction vectors, 3 - number of variables
0, 1, 0	the first prediction vector
0.1, 0.9, 0	the second prediction vector
0.3, 0.7, 0	the third prediction vector
0.4, 0.6, 0	the fourth prediction vector
0.5, 0.5, 0	the last, fifth prediction vector

Figure 1.49: Example of the PRD file and its description.

2.6.3 The AIN File Format

This format is a binary file and can be used only by aiNet. The AIN format stores additional information, compared to the CSV format. All possible settings of the coefficients and flags, all of the prediction vectors and the calculated error estimations are stored. The AIN format is recommended for general use. Files stored in this format may be translated into CSV later, by using the SAVEAS function.

Chapter 3

3. aiNet in the Future

We are continuously improving aiNet and we will be updating and improving the product.

3.1 The next releases of aiNET

aiNet will be supported with the following additional features:

- A *Print* command will be added to make a hardcopy of the charts.
- A fonts and colors dialog will be added to customize aiNet.
- More examples will be included.
- Some of the bugs you will report will be fixed.

3.2 The “More Hazy” Future

Better user support will be provided.

The major back propagation neural networks will be appended to the aiNet.

Some self organization algorithms will help build up the best possible model.

A library of the aiNet kernel functions will be available for the C++, C, FORTRAN and PASCAL programming languages all in ANSI standard (except PASCAL) to be moveable across different platforms.

Some special graphical tools for pattern recognition will be developed

3.3 You Can Help Us

We will continually collect your remarks, wishes and bug reports and we will try to take them into account whenever possible.

Please provide us with feedback concerning aiNet and the supporting documentation.

Chapter 4

4. All About Registration

This is a FULL version of aiNet. Nothing has been disabled in the trial period and there are no extra files in registered version. We think that releasing a full version is necessary and it enables you to fully evaluate aiNet before you decide to register it. This also means that we are relying totally on your honesty to register.

4.1 What are the Benefits of Registration ?

Registered users will be entitled to free aiNet upgrades, until (but not including) the next major release. Major upgrades are in whole numbers and minor upgrades are in .10 or .01 increments. For the new major releases a 50% discount applies to registered users.

Registered users have the chance to influence the features of future versions.

Registered users will have 100% support via e-mail or ordinary mail. This support is for correcting bugs in the software and manuals, but does not include advice on how to solve various problems using aiNet. We will also inform registered users about new issues of the aiNet or of our similar applications, but only if we are allowed to do so.

A 50% discount will be available to all registered users, when they will consult us about solving various problems using the aiNet. We are also available for contract work on the use of neural networks.

4.2 How to Register

If you use the aiNet for more than 21 days - three weeks - you are obligated to register it by paying the registration fee. To do so, fill the registration form, sign it and mail the payment to:

aiNet
Trubarjeva 42
SI-3000 Celje
Slovenia
Europe

If you are paying with a credit card, then you can also send the registration form via e-mail (AINET@SIOL.NET). Use the registration form template (ORDER.TXT), fill and drop it into your e-mailer. This will speed up our response.

As you will see from the registration form we provide two different ways to register. The first is fastest and cheaper; we will mail (or e-mail) you your registration text and code. After you receive the text and code, they need to be entered into registration dialog box. This will remove all

registration-demand messages and will explicitly register the aiNet to you (registration text will appear in the caption bar).

If you choose the second way, we will send you aiNet on a diskette. You will also receive the registration text and code, which can be used for next minor upgrades. Optionally, you may order also a hardcopy of the manual.

Currently, we accept three methods of payment: credit cards (preferable), money checks and payment to our bank account. Generally, there are no problems with credit card payment, but some problems may occur with money checks. For this reason, we prefer a credit card payment or payment to our bank account.

4.2.1 Registering the aiNet DLL library

There is an additional step in the procedure for registering the aiNet DLL library and the registration fee depends on what it will be used for.

Personal use

If you will use the aiNet DLL library strictly for your own personal use than no additional fee is required. (You must register aiNet, of course.) The only thing you must do is to sign the registration form and mail it to us. By signing the form you agree that the aiNet DLL library will not be used for any other purposes but your own and that the aiNet DLL library will not be sold, resold or distributed in any form to the third party. We will send your aiNet DLL registration code as soon as we receive the signed form from you.

Commercial use

We consider any other use but personal use described above as a commercial use. For the commercial use you must pay an additional fee of 200 USD; this means 400 USD registration fee for aiNet and the aiNet DLL library together. You can not register the aiNet DLL library only. By paying the aiNet DLL registration fee you are allowed to freely distribute the aiNet DLL files along with your application which you sell to third parties.

4.3 Disclaimer of Warranty

This software and documentation are sold "as is" and without warranties as to performance of merchantability or any other warranties whether expressed or implied. Because of the various hardware and software environments into which this program may be put, no warranty of fitness for a particular purpose is offered. You use the aiNet *entirely at your own risk*, and you supply it to your customers, friends, family or acquaintances *entirely at your own risk*.

In no event shall we (aiNet) be liable for any damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or other pecuniary loss) arising out of the use of or inability to use aiNet, even if we have been advised of the possibility of such damages. If these terms are not acceptable to you, then **DELETE** all the files from your disks *immediately and permanently*.

aiNet 1.xx REGISTRATION FORM (USE CAPITAL LETTERS, PLEASE)

Name: _____

E-MAIL Address: _____

Address: _____

Registration Name: _____ (Do not make it too long!)
(This name will appear on the aiNet's caption bar.)

How do you want to receive a registration:

I want to receive a registered version of the aiNet on a diskette.

I want to receive a registration card only (includes Registration Text & Code), but no diskette.

Send it to me via mail, e-mail.

Price Calculation:

Registration Fee: _____ Single user registration fee: US\$49. (Multiple users registration fee: Contact us)

Shipping & Handling: _____ Zero, if you do not order any manuals or diskette. Otherwise: Europe US\$5, other countries US\$10.

Commercial aiNet DLL reg. _____ US\$69 (Free for non-commercial and personal use. See below!) You will receive the source code, too.

Check payment FALUS: _____ US\$15. Banks take about US\$20 per check. For this reason we have to charge you estimated bank costs.

Cash payment BONUS: _____ DISCOUNT if you pay with the cash. US\$14 for the application registration and US\$24 for application & commercial DLL registration.

Total Cost: _____

Credit Card Orders: Please Check MasterCard Visa American Express
 Eurocard Diners Club

Cardholder's Name: _____

Address: _____

Credit Card Number: _____ Expiration Date: Month ____ Year ____

Credit card payments may be sent via mail or E-Mail!

Your Signature: _____

Check Payments: Make check payable in U.S. currency to: aiNet, Trubarjeva ul. 42, SI-3000 Celje, Slovenia, Europe
Do not forget the extra \$15 for bank costs.(See above – Price calculation)

Cash Payments: Enclose cash into a hard paper envelope. Be sure that the cash is not visible from outside. Embed the cash with the thick dark (black) paper once or twice. If you pay with the cash the application costs US\$35 and full registration (app. & commercial DLL) costs US\$94.

Mail the envelope to: aiNet, Trubarjeva 42, SI-3000 Celje, Slovenia, Europe

aiNet DLL library personal use registration:

By signing this form you agree that the aiNet DLL library will not be used for any other purposes but your own and that any product which will use the aiNet DLL library will not be sold, resold or distributed in any form to the third party. (This signature is not required if you pay Commercial DLL reg. fee.)

Our E-MAIL: AINET@SIOL.NET

Your Signature: _____

WWW: <http://www.ainet-sp.si/>