aiNet

A neural network Windows™ application,
Version 1.00, February 1995

PART ONE: USER'S GUIDE

PART TWO: BASICS ABOUT MODELING WITH THE aiNet

PART THREE: EXAMPLES

The authors welcome your comments, warnings, and critics on the program and this manual. Please write us to:

AINET,
Ales Krajnc, s.p.
Trubarjeva 42
SI-63000 Celje
Slovenia
EUROPE

or send email to:

AINET@IKPIR.FAGG.UNI-LJ.SI

Preface

In the beginning of 1992 Iztok Perus, one of our team members, began with the use of the neural networks. He tried to solve some simple problems in the field of civil engineering and all results were very promising. After he had moved to some more sophisticated problems, 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 the 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 was 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 cruel reality another. We could find no one, who would be prepared to finance the development of such an application. Finally, we decided to try it on our own and here it is: the aiNet application. We wrote this application in our spare time and we became more and more dedicated to the aiNet. One of us even quit his (very bad paid) job and the other will probably follow him.

Thus, the version 1.0 of the aiNet application is finished. But how to 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, 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. It will be a great help for you and for us. There are still so many ideas around and we want them to be build in the aiNet, but our resources (time & money) are exhausted. By registering this application you will help us to continue our work and to continuously improve the aiNet application.

Ales Krajnc, January 1995

Introduction

The aiNet application is a very powerful and a very simple tool for solving the problems which are usually solved with artificial neural networks (ANN). All possible tests we had run proved that the results obtained with the aiNet are at least as good as the results obtained with some other ANNs. Let us state some of the aiNet's features.

- The major attribute that distinguishes the aiNet from other ANNs is the analysis speed. Since the 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). Or in other words: 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 you can add some new data to neural network (or remove old ones), add some additional parameters (or remove old ones) and still get answers right away -- there is no time consuming learning phase.
- Our experiments show us that noisy data is the aiNet's favorite. If the data is just noisy, the aiNet will give you excellent results. When you have chaotic data (and you do not know that), then you can not obtain any solution, still the aiNet will assure you that something is wrong with the data.
- The aiNet provides you a way to estimate the rate of error in your prediction. If your problem is smooth, i.e. without noise, then this error will represent an estimation for the error in the predicted result. If you have noisy data, then this will represent an estimation for the noise around the predicted result. This means that an error estimation behaves locally.
- aiNet is very suitable to work with missing values in your data. In real life problems it is usually very difficult to find a perfectly assembled knowledge base -- there is always some data missing. The aiNet handles missing data automatically and you need not worry about how to represent such data.
- The aiNet's graphical user interface is very simple to use. It looks like a spreadsheet application and if you are familiar with any other spreadsheet, the aiNet would not present a problem for you. Almost everything is only a mouse click away, menus are simple and there are also speed buttons.

- On-line help is there for you. If you are stuck and do not know what to do just press F1 and the aiNet will help you find the way out.
- Several charts are also available. They represent the most natural way to estimate how good your problem data is. They tell you visually if your problem can be generalized and what kind of results you can expect.

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

- The aiNet is somehow not able to solve real time problems. When the aiNet is to calculate a single prediction, it scans through all the data in the "knowledge base" which takes some time. On my PC 486/66 it takes about 0.1 second to compute one prediction based on a knowledge base with 1000 samples each with 5 parameters. One tenth of a second is not very much, but for serious real time problems it is usually too much.
- The aiNet does not generate fixed weights, which can be later used for prediction purposes. The aiNet does compute weights, but the computing is done dynamically -- individually for each different prediction.

Manual Overview

This manual is divided into three logically separated parts:

- Part 1: Users Guide
- Part 2: Basics About Modeling with the aiNet
- Part 3: Examples

Part 1: “Users Guide” teaches you how to use the aiNet application. First it shows you how to install the aiNet application. Later it tells you how to enter the data, how to save it to or load it from the disc, how to perform your analysis, which buttons to push and which menu commands to select. To teach you all this, we will use two examples. The first example is the simplest possible and has besides educational no other practical meaning. The second one is slightly more complicated and has been selected to outline some features of the aiNet application. After that the file format is revealed, which the aiNet uses for saving its data. At the end of Part 1, the vision of the aiNet’s future is given. This will perhaps encourage you to register the aiNet and help us make this vision true.

The Part 2: “Basics About Modeling with the aiNet” tells you something more about modeling the problem you want to solve. It explains the major aiNet’s features in detail and shows how, why and when to use them. The explanation of these details is also based on simple examples. We can say: if Part 1 explains the technique, then Part 2 explains the philosophy of the aiNet.

The Part 3: “Examples” deals with several problems from various science areas. Every problem begins with the introduction section, 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.

Manual Conventions

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

`Monospaced type`	This font represents a text which you type or an onscreen text.
Italics	Italics are used to emphasize certain words, menu choices and indicate the aiNet terms.
Keycap	This font represents a particular key you should press -- 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 Authors

Currently, we have limited capabilities for a costumer support. When you want to get in touch with us, you can choose between two ways. You can either write to us a classical letter or reach us via Internet with an e-mail. Both addresses are listed below:

Post address: AI**NET,**
Ales Krajnc, s.p.
Trubarjeva 42
SI-63000 Celje
Slovenia
Europe

E-MAIL address: **AINET@IKPIR.FAGG.UNI-LJ.SI**

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 definitely can not be satisfied with such a poor customer support. We wish to improve it by way of a WWW server as a hot-line phone service. It is up to you, dear users. More registered users of the aiNet application means a better customers support.

Please, do not try to reach us by telephone. Our only line is overloaded even without your call! Thanks.

USER'S MANUAL
Part 1: Users Guide

Chapter 1

1. Getting started

In this chapter you will install the aiNet application and configure it into your environment. Afterwards you will launch it and solve a very simple problem. This will make you feel more comfortable with the aiNet and you will be ready for the next chapter.

1.1 Installing the aiNet

If you are an experienced user, then you have already installed and possibly l(a)unch☺ the aiNet and you can skip right to the "Solve Your First Problem". For the others we prepared some more information about the installation.

1.1.1 Hardware and Software Requirements

For successful work with the aiNet you need the following minimal configuration:

- a PC with the 386 processor or a better one,
- 4 MB of RAM on the motherboard (rather more, rather more, much more),
- about 3 MB of free hard disk space,
- Microsoft Windows 3.1 environment,
- VGA graphics card (the aiNet does not support Hercules mono graphics, although Windows do),
- a mouse compatible with the Windows,
- if you have 286 or 386 microprocessor, then a math co-processor is highly recommended,
- strong nerves

Please note that later versions of the aiNet may require different minimal configuration (5 MB of free hard disk space and very, very strong nerves, indeed, for example☺).

1.1.2 Installation

We do not know how you got our aiNet application. Did you download it from the Internet or a BBS or did you receive it from your friend? But we are sure it was compressed in some way and that you are experienced enough to decompress it and associate it with the Windows Program Manager. Yet, we would like to give some attention to the files: BWCC.DLL and AINET.INI.

Since the first one (BWCC.DLL) is often used by the applications developed with one of Borland compilers, you may already have this file on your PC. In this case it would be a good idea to move the latest version (Check the date!) of the BWCC.DLL file to the WINDOWS\SYSTEM directory. After that you may delete all old BWCC.DLL files.

The AINET.INI is a file where the aiNet application stores some necessary information. It will be automatically created the first time you will launch the aiNet application and it will be located in the WINDOWS directory. You do not need to have any access to this file. We simply wanted to remind you aware that this file exists.

1.2 Solve Your First Problem

Now it is time to solve your first simple problem with the aiNet application. This will give you some basic ideas of how the aiNet can 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 is trivial. You do not need a neural network to solve it, your biological network does this job much better. Anyway, here is the XOR data presented in a table form (see Table 1.1). Our job is to check if the aiNet can compute (predict) correct results, after it had seen these 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 the aiNet

We will model the XOR data in several steps; we will enter the data, normalize the data, enter some test values, run prediction and save the data. Let us now do one step at a time.

Entering the Data

Run the aiNet application. An empty window, with a menu and a tool bar appears on the screen.

In the *File* menu select the command *New* and a dialog box will pop up. The dialog will be asking you to enter the number of model vectors (data sets) and the number of parameters. As you see from the table above, we have 4 different sets (model vectors) and each model vector has 3 parameters (value A, value B and the Result). So, enter the numbers `4` and `3` and close dialog by pressing *OK* button.

Now the aiNet asks you for the Document type. It does not matter which one you select. You will learn something about that later.

If you do all that correctly, then you must see a grid of cells (4 rows and 3 columns).

Enter the values as shown in the Table 1.2. When asked to confirm some values, answer with the *OK* button.

You have entered the data now, but the aiNet does not know which parameters are an input and which are an output. (Currently, all are excluded - button E.) You can tell this by pressing the

appropriate button under each parameter name (currently named as P:1, P:2, P:3). Select button *I* for P:1, P:2 and button *O* for P:3.

Naming the parameters would be a good idea too. To do that, double-click on the parameter name (P:1). A dialog box pops out asking you to change some values. You do not need to change anything in this dialog box except the parameter name. Enter a new name, "`A`" for example, and close the dialog. Repeat this action for parameters P:2 and P:3 and name them as "`B`" and "`Result`", respectively.

Having done this, we have finished entering the data (model vectors) and we can proceed with 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. Note that the numbers in the cells have changed and you can see the normalized values instead the real ones. If you try to edit some cells, you will note that you can not edit normalized values, hence the term Lock in the menu command.

Running the Prediction

In this step we will see if the aiNet is able to correctly predict the result from the model vectors we have just entered and which are now normalized.

Select the *View*|*Prediction* command and a dialog box will pop up. It will ask you how many rows you want to have in the new prediction window it is about to create. This actually means how many prediction vectors (sets) do you want to have. Type number `5`, for example. A new window with 5 rows will be displayed. We will call this window a "Prediction View".

Now we can type some test data into the *Prediction View*. Here is the table of prediction vectors. Type in the numbers for parameters A and B. You have probably noticed that you can not enter the data into the output cells (parameter Result).

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	0.999
0.4	0.6	0.965
0.5	0.5	0.500

To calculate results select *Prediction*|*Calculate Prediction* command and in the third column appear the results of the calculation. As you can see from the table above, the results are written with three decimal digits, whereas the result in the *Prediction View* with has only got one decimal digit. To change this, double-click on the Result parameter and a dialog will pop up. In the dialog, enter number `3` beside the text "Number of decimals" and close it by pressing the *OK* button. Three decimals should appear in the Result column.

Now, you can type different combinations of input cells and calculate new results. Just do some typing and then select *Prediction*|*Calculate Prediction* command. It's easy, isn't it?

Saving the Data

To save the data select the *File|Save* command. The aiNet will display Save As dialog, where you can enter a file name and a location of the file. You can also choose a file format in the *Save File As Type* combo box. We prepared two different file formats: binary format and ASCII format.

ASCII format "Comma delimited (*.CSV)" stores all the model vectors and prediction vectors plus parameter names. If you want to use this format, you must denormalize model vectors first. This format is suitable for transferring information into other applications, like Microsoft Excel, Borland Quatro Pro, ... See also Chapter 2, aiNet File Formats.

Binary format "AINET files (*.AIN)" stores the data in the aiNet internal format. This format is not transferable, but stores more information. You also do not need to denormalize model vectors before saving. We recommend you to use the binary format, unless you want to use the data with some other application. If you want to do this, you can always select *File|Save As* command and save the data as an ASCII file.

1.2.3 Comments

If we take a look at the table with the prediction results, we can conclude that the 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 parameter for 0.2), the aiNet calculated a correct result. Even if we increased this change up to 0.4, the aiNet would still calculate a very 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. The aiNet in Details

In this chapter we will reveal all possibilities the aiNet may give to the user. Because the aiNet uses some specific commands, you may quickly scan through User's Manual, Part 2: Basics About Modeling with the aiNet before you continue with this chapter. Although this is not necessary, it might help you to better understand some of the topics in this chapter.

2.1 Four Different Views

Here comes a short presentation of the views, which are currently built in the aiNet. We must present them because some menu commands are related to the views.

When you working on the XOR problem, you used two views: *Model Vectors View* and *Prediction View*. Besides these two there are two more views: *Error Distribution View* and *Charts View*. As the names of the views reveal, they present your problem in different ways.

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 parameters. Usually you can not see all the model vectors and parameters at once, so you must do some scrolling. This view is automatically activated when you create a new neural network or you load an old neural network from the disk. You use this view to edit model vectors, to add some new or remove some old model vectors, etc. In this view you also normalize and denormalize model vectors.

Prediction View is used when you want to run some predictions. This view is very similar to *Model Vectors View* and allows you to take similar actions. You can edit the input part of the predictions vectors and then use the aiNet to calculate the output part. 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, the aiNet can besides prediction itself predict also the ratio of the error in the prediction result. Before the aiNet can do that, so called filtration and verification process must run over all model vectors. *Error Distribution View* shows the results of the filtration and verification. Like both views above, this view also uses spreadsheet-like window to present the results. This view is Read only view, which means that you can browse through the results only, but you can not change them.

Charts View is an extension of the *Error Distribution View*, because this view represents the same results as *Error Distribution View*, but in the graphical ways. This enables you to estimate these results visually. Two different charts are currently available. The first one is some kind of correlation chart. 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). The second chart is a bar chart. It shows the verification or filtration error of each model vector. Model vectors are enumerated on the horizontal axis and the error is shown on the vertical axis.

2.2 aiNet Menu

Since the aiNet is a user friendly application, the user commits all actions trough the menu commands or by pressing speedbar buttons. The aiNet commands are organized in menus and here we will try to explain them. The aiNet uses two kinds of menus: static and dynamic ones. Static menus are always present in the menu bar and are almost always accessible. Dynamic menus are view-depended. When the *Model Vector View* window is active, for example, then *Edit* menu and *Model Vectors* menu appear on the menu bar and they disappear when some other view becomes 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	Help on Help
Close	Error Distribution	Register	Arrange icons	About
Close View	Charts		Close All	
Save				
Save As				
Save Prediction				
Open Prediction				
Exit				

File Menu

New

Using this command you order the aiNet to open a brand new neural network. Before the aiNet lets you enter some data into a new neural network, a dialog box pops up. The dialog asks you about the number of model vectors and the number of parameters in the model vector. (You can always add or remove some model vectors or parameters later.)

After you have entered some numbers, the aiNet will ask you about the file format you want to use. Select the one you like and *Model Vectors View* window should appear on the screen. Now you can start entering the data.

Open

You use this command when you want to load some existing neural network from the disk. The aiNet will display the *File Open* dialog. There you select the file you want to load. If the file is loaded successfully, the aiNet will automatically open *Model View* and display model vectors from the file.

Close Neural Network

Close Neural Net command will close all opened views of the current neural network and delete the network from memory. Before the aiNet does that, it will remind you to (give you the last chance) to save the network.

Close View

The aiNet will close the active view of the current neural network. If this view is the last opened view, you will be asked to save the neural network to the disk.

Save

Save command will save the current neural network to the disk. It usually takes a moment or two for the aiNet to do this job. However, 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, but before this is done, a dialog is displayed. In the dialog you must enter a file name and a location of the file. You can also specify a file format you want to use. You can use either binary format (AIN extension) or ASCII comma delimited format (CSV extension). If you use the CSV format, then 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 *Prediction View* is active. *Save Prediction* allows you to save the prediction vectors separately from the model vectors. This is very useful if you want to have several different independent sets of prediction vectors. Predictions are always saved in the ASCII comma delimited format, but with a PRD extension.

Open Prediction

This command does the opposite from *Save Prediction* command. It is enabled only if *Prediction View* is active. It will ask you to select prediction file (PRD extension) you want to load. After you select a file, the aiNet will discard previous contents of *Prediction View* and will load it with the prediction vectors from the file.

Exit

Use *Exit* when you want to quit the aiNet application. Before the aiNet closes, it will ask you to confirm your decision. If you really want to quit the aiNet, you will be further asked to save the neural networks.

View Menu

This menu allows you to bring up the view you want to see. If the view has not been created yet, then the aiNet creates a window for the selected view and puts it on the top. If the view window already exists and is covered with other windows, then this command puts selected view window to the top. This rule holds for all four views.

Sometimes some of the views are 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

Comment Width

When you want to change the width of the comment column in *Model Vectors View*, then you can do that by selecting this command.

Each model vector can have comments. Although you usually do not need them, you will encouner occasions, when they will be pure gold. Imagine you are collecting model vectors from some experiment, which is time dependent. You put these model vectors into the neural network. Now you make analysis and you notice that there is something wrong with some of the model vectors. If these vectors do not have a comment, it is hard to tell when were they measured. But if they do have a comment, a date and time, then it is easy.

Fonts

In the aiNet version 1.0 this command is disabled. In later versions it will allow you to use different fonts in the views.

Register

If you decided to register the aiNet application, we will send you your unique registration code. You enter your name and code in a dialog box, which is invoked by *Register* command. After successful registration, the aiNet stops displaying registration messages.

Window Menu

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 job for the icons.

Command *Close All* closes all the views. While the aiNet does the closing, it also asks you if you want to save some of the opened neural nets.

Help Menu

The best think you can do with the help menu is to try it. Do so!

2.2.2 Dynamic Menus

Here are tables of dynamic menus. As we mentioned earlier, these menus are activated together with the views and you can see them only if the right view is active.

Model Vectors Menu & *Edit* Menu

Edit	Model Vectors
Add rows (model vectors)	Normalize + Lock
Add columns (parameters)	Denormalize + Unlock
Insert row/column	Normalization settings
Delete row/column	
Delete empty rows (model vectors)	
Undo	
Cut	
Copy	
Paste	

Add rows (model vectors)

When you have selected command *File|New*, the aiNet asks you for the number of model vectors and parameters. You have entered all model vectors and you are now running out of space.

Select this command to add some extra model vectors to *Model Vector View*. The aiNet will open a simple dialog. The dialog will ask you about the number of rows (model vectors) you want to add. Type the number and close the dialog with the *OK* button. The selected number of new rows will appear at the end of *Model Vector View*.

Add columns (parameters)

If you realized that your model needs more parameters, use this command to add some. A dialog will pop up asking you for the number of new parameters. All new parameters will have the exclude status turned on. (Exclude means neither input or output.)

Insert row/column

This command will be available only if you have previously selected a model vector (row) or a parameter (column). 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 (parameter) will be inserted before the selected column. The new parameter will have the exclude status turned on.

Delete row/column

This command will be available only if you have previously selected a model vector (row) or a parameter (column). Before the selection is deleted, the aiNet asks you to confirm the command.

Delete empty rows (model vectors)

This command deletes all model vectors with no entries. It sometimes happens that you have initially created too many rows or you may have added too many rows later. Nevertheless, it would be a good idea to delete such empty rows. This will not only free some memory, but it will speed up the computation as well.

Undo, Cut, Copy, Paste

Undo works within *Edit* window only, which is located in the speed button bar. The same is valid for the *Cut*, *Copy* and *Paste* commands.

Future releases of the aiNet application will probably further implement these commands.

Normalize + Lock

Before you can do any computation with the aiNet, you must normalize the model vectors. After the normalization is done, the aiNet will enable a lot of commands, which you will need to perform calculations. This command also locks the model vectors - you can not edit them.

Denormalize + Unlock

This command reverses normalized model vectors back to their original values. It also unlocks them and thus allows you to edit them. But do not forget that when model vectors are unlocked, you can not perform any calculations.

Normalization settings

Since this command has a major influence on the results and since all views enable this command, we will explain it later in the text. See Three Most Important Commands.

Prediction Menu & *Edit* Menu

Edit	Prediction
Show local error	Calculate prediction
Hide local error	Penalty settings
Add rows (predictions)	Normalization settings
Insert row	Local Error Settings
Delete row	
Delete empty rows (predictions)	
Undo	
Cut	
Copy	
Paste	

Show local error

The aiNet can besides prediction calculate also an estimation for the error (or the noise rate) for every single prediction result. *Show local error* command displays these estimations in the *Prediction View* window. Please note that if you really want to see these estimations you must calculate error distribution first. If error distribution is not calculated, then only empty lines are displayed.

Hide local error

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

Add rows (predictions)

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

Insert row

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

Delete row

A row must be selected, before this command can be used. It will delete the selected row. You will have to confirm this command.

Delete empty rows (predictions)

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

Undo, Cut, Copy, Paste

Undo works only within the *Edit* window, which is located in the speed button bar. The same is valid for the *Cut*, *Copy* and *Paste* commands.

Calculate prediction

When you have entered all input values in the prediction vectors, you can calculate the output values selecting the *Calculate prediction* command. In a moment or two results will appear in the output parameters of the *Prediction View* window. If the error distribution was calculated before, 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

Penalty settings is one of the most important commands of the aiNet application. It will pop up a dialog box, where you will set and edit penalty coefficients and flags. All this will be explained later in the text. See Three Most Important Commands.

Normalization settings

Since this command has a major influence on the results and since all views enable this command, we will explain it later in the text. See Three Most Important Commands..

Local error settings

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

Error Distribution Menu & *Edit* Menu

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

Show difference & Show predicted value

As already told, the 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 can 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 the aiNet, Chapter 3.

Calculate error distribution

With this command you engage the calculation of error distribution. In a couple of moments results will appear in the output parameters of the *Error Distribution View* window. If there are many model vectors, the calculation of the error distribution can take a considerable amount of time. (**NOTE**: When you double the number of model vectors, then the calculation will take four times longer.)

Global error report

If you want to see global error estimates (root mean square error = RMS) then select this command. A dialog will be popped up, where the total RMS error[##] will be shown, as well as the RMS errors for single output parameters. See also Part 2: Basics About Modeling with the aiNet, Chapter 3.

Penalty settings

Penalty settings is one of the most important commands of the aiNet application. It will pop up a dialog box, where you will set and edit penalty coefficients and flags. All this will be explained later in the text. See Three Most Important Commands.

Normalization settings

Since this command has major influence on the results and since all views enable this command, we will explain it later in the text. See Three Most Important Commands.

Local error settings

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

2.2.3 Three Most Important Commands

One of the features of the aiNet application is that the aiNet uses only one coefficient, which has major influence on the results. This is true in almost all cases. Besides this coefficient, there are also a few commands, which effect the results. We shall now explain the usage of all this commands in more detail.

[##] 'total RMS' means one error for all output parameters

Normalization Settings

When you select this command a dialog pops up. It allows you to choose between two kinds of normalization procedures:

- regular
- statistical

Before the aiNet can perform any kind of calculation, all model vectors must be normalized. This means that some real physical dimensions will be transformed to something else. As we mentioned earlier, two normalization methods are available. Let us explain them.

If you choose a regular type of normalization, then the aiNet will perform the following actions during normalization.

1. Scan trough all model vectors and find the largest (maximum) and the smallest value (minimum) for each parameter.
2. For each parameter 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 somehow different. Here are the steps that the aiNet will perform during statistical normalization:

1. Run through all model vectors and calculate an average and a standard deviation for each parameter respectively.
2. For each parameter 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.

How to choose the right kind of normalization? Here is the rule we use (it is not an ultimate rule):

Use regular normalization if the data in model vectors is already normalized in some way -- if you know for sure that all values will fall within some physical limits.

When you can not say for sure that all the values will be between some limits, use statistical normalization.

In most cases the normalization type does not reflect significaly on the results. Please note, that 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 and we are sure you will use it a lot. When you select this command a dialog appears on the screen. The dialog asks you for the value of penalty coefficient and also lets you select a type of the coefficient.

Choosing the right value of penalty coefficient usually solves the problem. In the Introduction we already describe that there is only one optimal value. We also told you that optimality curve at this optimal value is shallow. This means that you can get good results even if your value is not so close to the optimal one.

We can give you only informative instructions for selecting the right value. Our experiences can be summarized in the following simple rules:

- The more model vectors you have, the smaller the optimal value is.
- The more parameters you have, the larger the optimal value is.
- The more noise your data has, the larger the optimal value should be.
- The best thing you can do is try a few values. Do not forget to try some stupid values also.
- Small penalty value leads to overfitting (overtraining) and
- Large penalty value leads to underfitting (overgeneralizing).

The dialog which pops up gives two possible choices for the penalty coefficient type:

- static
- dynamic

What do these choices mean?

The static coefficient type means that penalty coefficient value will be the same all over the hyper-dimensional space. Use this type when your model vectors are approximately uniformly (regularly) distributed. If your data is noisy, then this selection is also recommended.

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 prediction vector, then penalty coefficient decreases -- fits little more. However, if model vectors are rare in the surroundings of prediction vector, then penalty coefficient increases -- generalizes little more. Use the dynamic coefficient type, when model vectors are grouped in clusters and are not uniformly distributed over hyper-dimensional space.

Please note that switching from the static to the dynamic type (or opposite) reflects also on an optimal penalty coefficient value. The dynamic type usually leads to smaller values.

Error Distribution Settings

So far we have been using two terms: an error distribution and a local error (or local error estimation). These terms are related and have the same background.

Error distribution is used together with the terms filtration and verification. Actually, the error distribution is the result of filtration and verification. We named it distribution, because filtration and verification calculate an error for each model vector. When we know the error for each model vector, we also know how error is distributed over all model vectors, hence the term distribution.

Local error estimation is associated with prediction. If we know the error distribution and we run the prediction process, then we can predict not only the output parameters, but also the error for every single prediction. Because such an error prediction can behave locally (it is not a constant in the hyperspace), we use the term local error. Since error distribution is always noisy, this error prediction can not be very accurate, hence the term estimation.

Using the *Error Distribution Settings* command you control the calculation of the error distribution. You can turn the error distribution on or off by selecting between these two possibilities:

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

You can also select which type of local error you want to use. There are two types:

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

Use an unsigned (absolute) type if you have noise in your model vectors. When you know that your problem is smooth and without any noise, then you may try with signed local error type.

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

2.3 Solve Your Second Problem

In the previous section we became familiar with all commands in the aiNet application. Now we will use some of them in our second example. This example is slightly more complicated as the first one, yet simple enough to be understood.

2.3.1 About the Hole in a Square Problem

Imagine a two dimensional square region, which has a hole in the middle. We put this square region into a coordinate system, so that every point in the region can be located knowing point's coordinates. For our example, we selected a two unit square and a hole with 1.4 units in diameter. This is shown in the Figure 1.1.

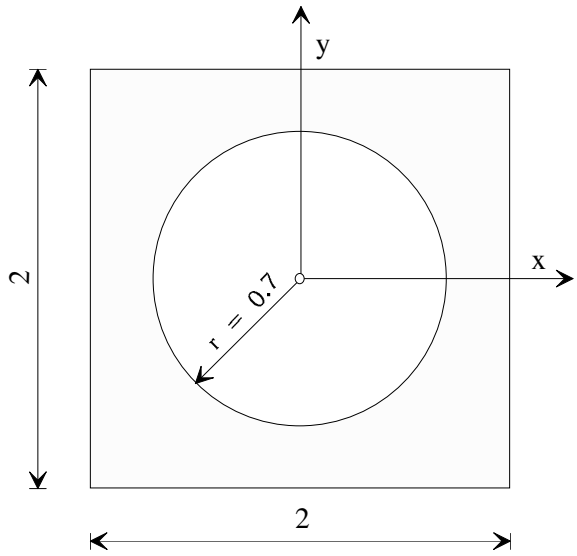

Figure 1.1: The hole in a square problem.

Let us select one random point in the square region. Knowing the point we can easily determine whether this point is in the hole or not using a simple 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

Now we would like to train the aiNet to do the same job; if we give it a point, then the aiNet must tell us where this point lies -- in the hole or outside the hole.

Before we can expect some answers from the aiNet, we must provide it with some training examples -- model vectors. Since there can be a few hundreds of model vectors, we will not type them into a *Model Vectors View*, but rather generate them.

2.3.2 Generating Model Vectors

The simplest way to generate model vectors is to use one of the spreadsheets. We used a random generator function and we generate 200 random points in the square region. For each random point we calculated the point's location. If the point lied in the hole we assigned it value 1, but if the point lied outside the hole, we assigned it value 0. Finally, we ended up with a table of 200 random model vectors. Each model vector has three parameters: x coordinate, y coordinate and a flag indicating the point's position.

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

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 1.2.

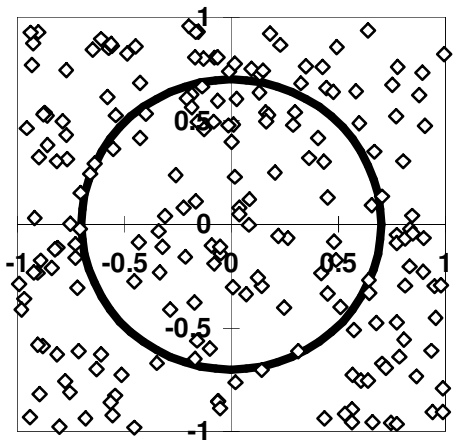

Figure 1.2: The hole in a square problem - database.

All these model vectors are saved in the file named HOLE.CSV. To learn something more about file formats the aiNet uses, see aiNet's File Formats. To open and load this file, select the *File|Open* command and find HOLE.CSV file in the dialog. Close the dialog with the *OK* button.

Now, we have sample points -- model vectors loaded into the aiNet. Before we can carry out a prediction, we must perform some modeling on our model vectors.

2.3.3 Modeling with the aiNet

Modeling with the aiNet is a simple iterative procedure. First we normalize model vectors, then we iteratively select an optimal penalty coefficient and finally we can calculate some predictions.

Normalization

Since we have uniformly distributed coordinates we can select regular normalization. To do so, Select *Model Vectors*|*Normalize* settings command and turn the option named regular on.

To normalize model vectors, select *Model Vectors*|*Normalize* command. When this is done, switch to the *Charts View* window. To do so, select the *View*|*Charts* command.

Optimal Penalty Coefficient

Select the *Error Distribution*|*Penalty settings* command to display a dialog. Before we choose some value for the penalty coefficient, it is a good idea to select the penalty type. Since there are a few clusters of points and also a few white areas in the Figure 1.2, we decided to choose the dynamic penalty type. (Note, that in this case this decision is not very important.) When the penalty type is set, we can select some value for penalty coefficient.

The best way to find the optimal value for penalty coefficient is to run filtration and verification and to observe the RMS verification error. The value, which will give 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

Let us try to find the value which will minimize the verification RMS error. We selected several values for penalty coefficient and we ran filtration and verification for each value. If you want to do the same just 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 1.3.

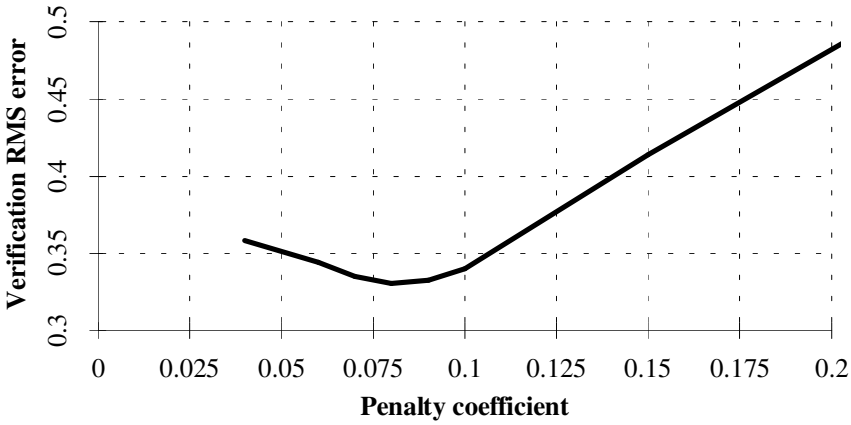

Figure 1.3: Penalty coefficient via verification RMS error.

As we can see from the results that a good estimation for the penalty coefficient would be 0.08. We can also see that the optimality curve is really shallow at the minimum.

Verification of the Model

We found the optimal value of penalty coefficient and we can stop with modeling at this point. We know that we can do that from the experience in the past. But you must feel strange, especially if you have some experience with back-propagation networks. To convince you that we have really got good results we will test our model with new data which the aiNet never saw before.

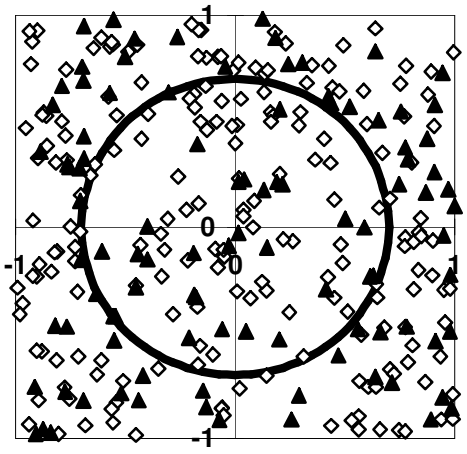

Figure 1.4: The hole in a square problem - test example.

To do so, we will generate new random 100 points, which will be scattered over our square region. For these points we will calculate exact results and we will also ask the aiNet to predict results from the new points. Then we will compare these results.

We generated new 100 prediction vectors and we put them into the HOLE.PRD file. The new points (prediction vectors) are shown together with model vectors in the Figure 1.4.

To calculate prediction for new prediction vectors, bring the *Prediction View* window to the front. To do so, select the *View|Prediction* command. Now load the HOLE.PRD file into *Prediction View*. You will do this by selecting the *File|Open Prediction* command. In a dialog, which has popped up, select the HOLE.PRD file and close the dialog with the *OK* button.

Now check if the penalty settings are correct. Select the *Prediction|Penalty Settings* command and check if the dynamic penalty type is set and if the number 0.08 is entered into the edit box. Close the dialog and select the *Prediction|Calculate Prediction* command.

In a few moments, the aiNet will calculate predictions and will display them in the *Prediction View* window. Let us remember; if a point lies in the hole, then we assign it value 1 and if the point lies outside the hole, then we assign it value 0. The aiNet can not give you as sharp answers as mathematics can, instead of it will give you some value between 0 and 1. We will assume that if the result for a given point is below 0.5 that point lies outside the hole and if the result is above or equal 0.5, then the point lies in the hole. To see no decimal part in the third column, double-click the parameter name "Result" and select 0 in the *Number of decimals* edit box.

And what did our prediction show? We know the exact results and we can see the prediction. If we compare both results, we can see that the aiNet failed to predict three from 100 points. This means a 97% success[++] . Not so bad for only 200 model vectors. Furthermore, we can switch to *Error Distribution View* and count 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 and in addition it shows also that you usually do not need a test set to model neural net.

We repeated this prediction test with some different values for the penalty coefficient and now we present the results 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

Now we will solve the same problem as before with only one change. We will add some noise to the third parameter "Result". This noise will be added to all 200 model vectors. The noise will be simulated with the gaussian distribution, where the mean value is 0 and standard deviation is 0.5. This is quite a noise considering that points in the hole have value 1 and points outside the hole have value 0.

[++] One hundred predictions vectors is just a small test set. To be more sure in the precision of the aiNet prediction, we should build a larger set.

Model vectors for this noisy example are stored in the HOLE-NS.CSV file.

Optimal Penalty Coefficient

Using the same procedure as in the previous example, we found out that the optimal value for the penalty coefficient should be 0.16.

Verification of the Model

For the verification we used the same prediction vectors as in previous example. Specifying 0.16 for the penalty coefficient and calculating the prediction we obtained surprisingly good results. The aiNet failed in only four cases which means a 96% success.

We repeated this prediction test with some different values for the penalty coefficient. The results are collected in a table 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

We may ask ourselves how to improve our results. There are many possible ways to do that. Let us reveal two possibilities.

One of the simplest ways of improvement would be adding new model vectors to the existing ones. In reality you do this by collecting additional information about your problem. You then transfer the information into new model vectors or maybe even into some new parameters. This is the most common situation.

If you have unlimited source of model vectors and your problem is smooth (as in our smooth example) then you can improve results using the following algorithm.

1. Put a small number of model vectors into the neural network knowledge base.
2. Get some new model vectors, which are not in the base and copy them into prediction vectors.
3. Run 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 base.
5. Repeat the steps 2, 3 and 4 until you obtain the desired precision.

We are sure that this way of improvement works very good in our smooth example, but we are not so sure, if it would work so in our noisy example. (We actually did not try this out.)

2.3.6 Final Comment

We would like to emphasize one very important thing about neural networks. You can have the best possible neural network application in the world, but you will still get bad results if your data

(model vectors) is bad. This means that you must mainly concentrate on your model vectors and ask questions about them. Do I use too many parameters in a model vector or maybe too few. Should I add some new model vectors or should I take some away. Are my model vectors complete, where did I get them, are they compatible and so on. If you can give answers to these questions, then you will find the problem's solution much more quickly as you would find it simply by changing the number of neurons, hidden layers or by tuning a couple of parameters or penalty coefficients.

The aiNet, like any other neural network application, is just a tool. Its major feature is simplicity and its orientation toward modeling, so you can really concentrate on your model vectors - data and on the questions we mention above.

Once, one friend of ours worked on some problem. He did not know much about this problem so he collected all possible data he could get (and he got a lot). He put all these data into the model without asking himself any of the questions mentioned above. He ran a poor neural network over and over again and he never got a good result. One day his 5 year old son visited him at his office. When his father had explained his problem to the kid, the kid spoke to him "Shit in! Shit out!" and he solved the problem.

2.4 aiNet's File Formats

The aiNet spreadsheet-like editor is a great tool for entering model vectors. However, you must type-in your model vectors -- you can not avoid that. But many times you get model vectors in the electronic form. Retyping all the model vectors in the aiNet would be a waste of time. The same is true, if you want to use the results of prediction in some other application for further processing. For this reason we build in the aiNet two ASCII based formats: a CSV and a PRD format which are used for storing model vectors and prediction vectors, respectively.

2.4.1 The CSV File Format

When you get model vectors in an electronic form, it is much easier to convert your model vectors into the CSV (comma delimited) format, which is understood by the aiNet. We usually perform conversions with one of the spreadsheet applications. After you have converted model vectors into the CSV format, three additional lines must be inserted in the top of the CSV file.

Here is how the CSV file must look like, to be properly understood by the aiNet.

row 1:	`P_COEF, RES, RES, NMV, NP, RES`
row 2:	`PAR_NAME_1, PAR_NAME_2, ..., PAR__NAME_NP`
row 3:	`PAR_ST_1, PAR_ST_2, ..., PAR_ST_NP`
row 4:	`MV_11, MV_12, ..., MV_1NP, Comment1`
row 5:	`MV_21, MV_22, ..., MV_2NP, Comment2`
rows :	`...`
row 3+NMV:	`MV_NMV1, MV_NMV2, ..., MV_NMV.NP, CommentNMV`

Here is the explanation of abbreviations we used:

`P_COEF`	penalty coefficient value,
`RES`	reserved field. It should be set to 0 (zero).
`NMV`	number of model vectors,
`NP`	number of parameters,
`PAR_NAME_j`	the name of the *j*-th parameter,

`PAR_STATUS`	the status of the *j*-th parameter,
`MV_ij`	the value for the *j*-th parameter in the *i*-th model vector,
`Commenti`	optional comment for the *i*-th model vector.

For penalty coefficient you can specify any value, which is greater than zero. Later, when the file will be read by the aiNet, you can change this value as much as you like. The type of penalty coefficient as well as the normalization type and the local error estimation type must be set within the aiNet application, after the file was read.

All reserved fields must be set to zero.

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

The comment is an optional part of each model vector. You can put into a comment anything you like; usually some useful information about the model vector, like: Where did you get your comment vector?, when?, why? ...

If you have some missing values in your model vectors, then do not write any value - just write a comma. Here is an example: `1,2,3,,5` in the CSV file means 1,2,3,missing value,5.

Let us see a simple example of the CSV file (see Figure 1.5). The file for the XOR problem, which was presented in the second subsection of Chapter 1, is shown below.

`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 parameters: “A”, “B” “A XOR B”, respectively
`I , I ,O`	status of parameters: input, input, output
`1,1,0`	the first model vector
`1,0,1`	the second model vector
`0,1,1`	the third model vector
`0,0,0`	the last model vector

Figure 1.5: Example of the CSV file and its description.

2.4.2 The PRD File Format

This format is also the CSV format, but instead of the description of parameters and model vectors it stores the prediction vectors. The PRD format is a little more simpler, too. This is its structure:

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`

Where abbreviations mean:

`NPV`	number of prediction vectors,
NP	number of parameters,
PV_ij	the value of *j*-th parameter of the *i*-th prediction vector.

To enable loading prediction vectors from the PRD file, the number of parameters (NP) in the PRD file must be equal to the number of parameters in model vectors. This means that whole (input part+ouput part+excluded part) prediction vectors must be put in the file, not only the input part. You can assign any values to the output part because these values will be overwritten during the prediction process.

Now let us have a look at an example of the PRD file. We will present predictions vectors of the XOR problem, which are shown also in Figure 1.6 below.

`5,3`	5 - number of prediction vectors, 3 - number of parameters
`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 1.6: Example of the PRD file and its description.

2.4.3 The AIN File Format

This format is a binary file format and can be used only by the aiNet application, which in the other words means, that the AIN format is not transferable and this is its disadvantage. But AIN format stores some additional information: all possible settings of coefficients and flags, all prediction vectors and calculated error estimations. Thus the AIN format enables you to start working without any loss of data. We recommend you to use the AIN file format for saving your data. If you later want to convert the AIN format to the CSV format, you can always select the *File|Save As* command and save the data as appropriate.

Chapter 3

3. The aiNet in the Future

This is the first version of the aiNet application and one of our greatest wishes is that it will not be the last. There are many things we want to add, improve, rewrite ... and here we would like to announce some of the new features, which may be released in new versions.

3.1 The “Coming Soon” Future

We plan to develop a new 3D chart type - a surface chart. This chart will allow you to select three parameters. The first two parameters will be input and the last one the output parameters. You will also specify the range for the input parameters. The aiNet will then calculate the predictions of the third (output) parameter by changing the input parameters over the specified range. The surface then will be represented in the chart either in a real 3D view or in a top down view - isoline view.

We will add a Fibonacci (or maybe a Golden Section) Search, which will help you find an estimation for the optimal penalty coefficient value.

A calculation of basic statistical information will be added to help you see what kind of data you are dealing with.

We are testing now a new dynamic type of penalty coefficient and as soon as the testing will be done, it will be built in the aiNet.

The *Cut*, *Copy* and *Paste* commands will be implemented to speed up the data transfer to and from other applications, mostly spreadsheets.

The *Print* command will be added to make a hardcopy of the charts.

The font and colors selection dialogs will be added to customize the aiNet to your needs.

Some more examples will be included in the example part of the manual.

Some of the bugs you will report will be fixed.

3.2 The “More Hazy” Future

A better user support will be provided.

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

Some self organization algorithms will help you 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 the ANSII standard (except PASCAL) to be moveable across different platforms.

Some special graphical tools for pattern recognition will be developed and various other things yet to be discovered.

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 it will be possible.

If you have time, please send us your answers to the questions below:

1. What problems have you encountered while using the aiNet application?
2. What features of the aiNet do you especially like?
3. What features of the aiNet you do not like or maybe you do not understand?
4. What features of the aiNet need to be improved ? How do you think they should be improved ?
5. What needs should be added to the package to make it more useful to you?
6. What do you think about the manual we wrote?
7. Feel free to write anything else you like.

Chapter 4

4. All About Registration

We did not cripple this program in any way. This means that this is a FULL version of the aiNet application. Nothing has been disabled and there are no extra files. There is only a slight difference in the manual. If you order a hardcopy of the manual, there will be some pictures included, which you can not see in this manual. These pictures are in a bit map format and they consume an enormous amount of memory -- a few megabytes and you know how long do it takes to download a few megabytes from an Internet server. Only for this reason we put them out of the manual.

We think that releasing a full version is necessary and it enables you to fully evaluate the aiNet before you decide to register it. This also means that we are relying totally on your honesty to register.

4.1 What Is Your Benefit

If you become a registered user, you will be able to receive free upgrades to all future shareware releases of the aiNet application, till (but not including) the next major release. We will number major upgrades in whole numbers and minor in .10 or .01 increments. As we already said, the minor upgrades will be free. For the new major releases we will give you a 50% discount if you are a registered user.

You will also have a chance to have an influence on the features of the future versions, the same is not guaranteed for unregistered users.

Registered users will have 100% support via e-mail or ordinary mail. This support is for correcting bugs in the software and manuals and does not include advice on how to solve various problems using neural network. 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 application. 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
Ales Krajnc, s.p.
Trubarjeva 42
SI-63000 Celje
Slovenia
Europe

If you are paying with a credit card, then you can also send the registration form via e-mail (AINET@IKPIR.FAGG.UNI-LJ.SI). Just take a registration form template (ORDER.TXT), fill and drop it into your emailer. This will speed up our response.

As you will see from the registration form we provide two different ways for the registration. The first way is faster and slightly cheaper; we will mail (or e-mail) you your registration text and code. After you receive the text and code, you will enter them into the 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 the aiNet application 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 two methods of payment: credit cards (preferable) and money checks. 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.

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 s.p.) 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 the aiNet application, 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 1.0 REGISTRATION FORM (USE CAPITAL LETTERS, PLEASE)

Name:_____
E-MAIL Address:_____
Company/Address: _____
_____

Registration Name:_____(Do not make it too long!)
(This name will appear on the aiNet's caption bar. The registration code, you will receive, will be calculated on the basis of the Registration Name)

How do you want to receive a registration:
[ ] I want to receive a registrered version of the aiNet on a diskette.
Mail it to me in a [ ] 5.25" [ ] 3.5" diskette.
[ ] I want to receive a registration card only (includes Registration Text & Code), but no diskette.
Send it to me via [ ]mail, [ ]e-mail.
I order _____ copy(s) of aiNet manual at the price of US$10 for each copy.
This makes US$_____ in total.

Price Calculation:

Registration Fee:_____	Single user registration fee: US$200. Multiple users registration fee: 2-5 users US$160 per user, 6-10 users US$130 per user, 11 or more users US$100 per user.
Manuals: _____	Fill in the total from manual order (above), or write zero if you do not order any manuals.
Shipping & Handling:_____	Zero, if you do not order any manuals or diskette. Otherwise: Europe US$4, other countries US$9.
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!

Signature: _____ (for Mail orders)

Check Payments: Make check payable in U.S. currency to:

AINET
Ales Krajnc, s.p.
Trubarjeva 42, SI-63000 Celje
Slovenia, Europe

Internet:
ainet@ikpir.fagg.uni-lj.si

[ ] I want, [ ] I do not want to be informed about new issues of the aiNet.

Thank you for registering the aiNet application!

USER'S MANUAL
Part 2: Basics About Modeling with the aiNet

Chapter 0

1. Introduction

This part of User's Manual will explain you some basic features about modeling in general and how to use this knowledge in the aiNet application. The authors' wish was to make a computer program which allows a simple work on one side, and a great efficiency on the another. Therefore they introduced a few new concepts, which allow a new, so to say, a more natural view on practical problems in different (science) areas.

The theoretical basis will not be described in details. Moreover, we will try to avoid them through simple practical examples which will help users to accustom slowly to a new way of thinking. The basic idea of the aiNet comes from artificial intelligence, or more precisely, from its special area - neural networks. While such tools work as "black boxes", some assistant tools were added to improve the efficiency of the aiNet. These tools give, among the others, different mathematical measures, which correlate with some basic statistical measures. Therefore the efficiency of the model can be defined mathematically, the efficiency of different models can be compared together; reliability and/or the quality of the solution can be compared with the existing solutions. Finally, there are also some ideas (unfortunately, they are not included in the program yet), which allow a simple explanation of the predicted results.

Authors' expectations are, that the aiNet will be able to solve most of the problems in various areas, where there are sufficient data on measurements and/or sufficiently strict descriptions (not to "soft knowledge") of phenomena. Using the aiNet program we can avoid a lot of problems around the selection of different coefficients, which define the learning of usual artificial neural networks and also the problems of local minima (see, for example [1], [2], [3], [4]). The aiNet can be used as a filter in data preparation for other neural networks. Further it can be used for the

---

[1] Ramirez, M., R. & Arghya, D., **A faster learning algorithm for back-propagation neural networks in NDE applications**, Proceedings of the 2nd International Conference on AI, pp. 275-283, 1991.

[2] Samaad, T., **Backpropagation Improvements based on Heuristic Arguments**, Theory Track, Neural and Cognitive Sciences Track, International Joint Conference on Neural Networks, Vol. 1, Washington, D.C., 1990.

[3] Johansson, E., M., Dowla, F., U. & Goodman, D., M**., Backpropagation learning for Multi-layer Feed-forward Neural Nets Using the Conjugate Gradient Method**, Lawrence Livermore National Laboratory, 1990.

[4] Shanno, D., F., **Conjugate Gradient Methods with Inexact Searches**, Mathematics of Operations Research, Vol. 3, pp. 244-256, 1978.

analysis of problems itself, and for many other purposes. We are sure that the users will obtain a lot of answers to their questions and learn more about the phenomena under consideration.

NOTE: All examples shown in the manual are calculated with the aiNet. They are not fictitious, moreover, they represent a real capability and the efficiency of the aiNet!

Chapter 1

1. Modeling a phenomena

1.1 General

A tendency to master the events in every-day-life demands foreseeing - prediction of natural phenomena. We want to predict phenomena with different mathematical models. Modern science enables such models with its analytical approach. Reliable models of different phenomena are strongly connected with the events in the developed world: a competitive struggle on the market and material progress. Let us just try to analyze money market and insurance on the one side and health service and optimal solutions in technical disciplines on the other side ...

A reliable treatment of natural phenomena is based on measurements and the description based on relations between the observed results. From the theoretical point of view, the relations are most appropriately specified in terms of abstract mathematical models representing mathematical laws. But from the practical point of view, simulated analog models based on electronic devices are sometimes more convenient. The aim of this part of manual is therefore to introduce a generalized interpretation of a description of phenomena which can be realized in various information processing systems. The aiNet simulates one of such information processing systems.

It should be noted that in the process of modeling phenomena we want to ***describe*** **extremely complex** *reality* on the one side with **simple**, usually **very simplified**, *abstract mathematical models* on the other side (see Figure 1.1).

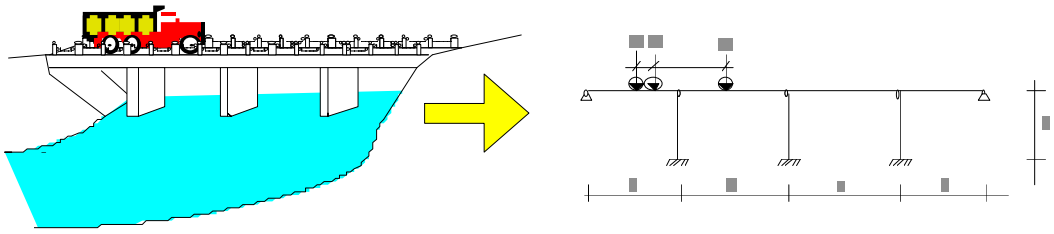

Figure 1.1: Description of the reality with the abstract mathematical model.

1.2 Describing a Phenomena

Let us have a look at a simple example of a natural phenomenon, which is graphically presented in Figure 1.2. These are two points in Cartezius coordinate system. They represent results of the

measurements - air temperature as a function of the day hour (e.g. temperature at 8:00 a.m. was 5.3°C, at 12:00 was 13.1°C).

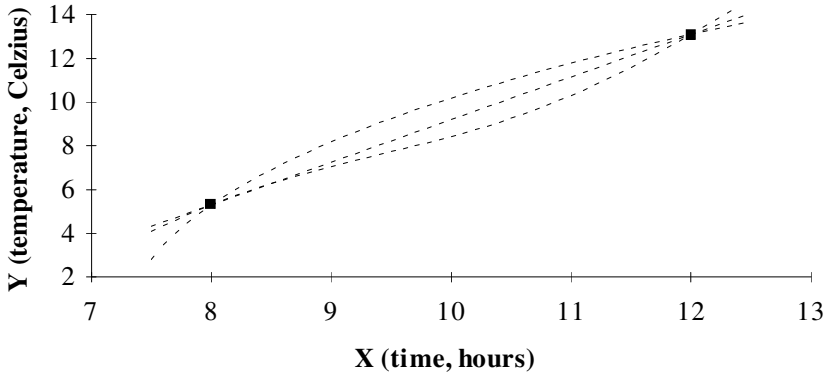

Figure 1.2: Graphical representation of a phenomenon: changing of the day-air temperature.

And what is the model which describes the changing of the day-air temperature? While we have only two measurements - two points uniquely define only straight line - the model can be linear. But, besides both measurements we also have supplementary, qualitative information about a phenomenon (the temperature has usually a low value in the morning, the highest value approximately at noon, and then it slowly decreases till the evening or till the night). According to this we can conclude that the linear model is not sufficient for the description of phenomenon (which is basically non-linear). Therefore, here comes the most important rule in the modeling process:

ALWAYS TRY TO COLLECT ENOUGH DATA WHICH ARE NECESSARY TO DESCRIBE THE PHENOMENON PERFECTLY (SUFFICIENTLY)!

The present example of a natural phenomenon is only two-dimensional in the mathematical sense. The rule might sounds trivial in this case, but practice shows that there are many, many real problems, where we want to model something for which we do not have enough appropriate data. This is not the problem of the modeling itself but the problem of acquisition of knowledge about phenomenon.

The problem is solved with acquisition of additional information of the phenomenon: the results of new measurements or additional expert's knowledge. New measurements in our case give the situation, presented in Figure 1.3. The day-temperature changing is shown at proportional time steps from the morning till the evening. Obviously, the phenomena is non-linear. As Figure 1.3 shows, the new data are sufficient for satisfactory description of the phenomenon.

In practice, in some (rare) special cases, the situation where there are too many data can also appear. Such cases do not have influence on the modeling of the phenomenon, only the computing time might become time consuming. The model becomes less efficient for this reason, moreover, in some special case, e.g. in real time applications it might become even useless.

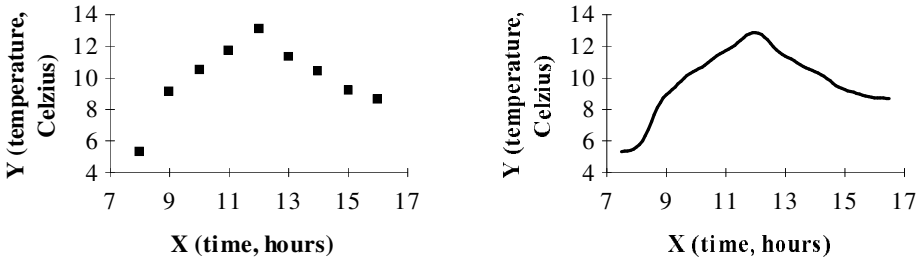

Figure 1.3: Graphical presentation of the day-temperature changing at proportional time steps and its mathematical model.

The solution of the problem with too many data involves a very complicated procedure. While it is not included in the aiNet yet and is not important for the further understanding, we will omit the description of the method. It should be mentioned that the method corresponds to the well known self-organization process of neurons [5.]. The number of data (samples) compresses to a smaller number of data (samples), which are representatives of the original data. This can be done in an optimal way where the new data are not just the extracted data out of the original data, but new formed abstract data [6].

1.3 Noisy Data

There are many other problems in the practice. If the day-temperature is measured with three different instruments in accidental time steps (such data reflect real life cases), we obtain the situation as shown Figure 1.4. A phenomenon where simultaneously measured characteristics (with different instruments) gain different values is called noise. This is not a characteristic of the measured phenomenon only.

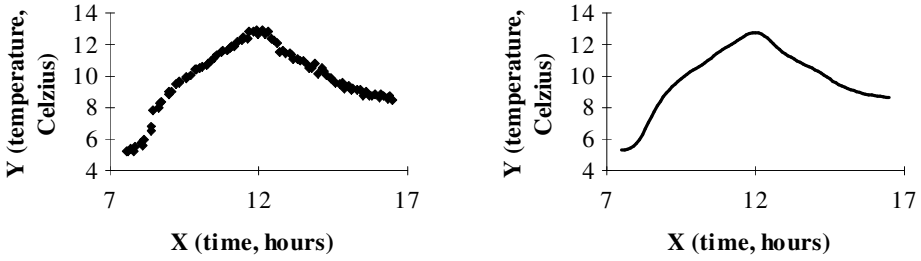

Figure 1.4: Graphical presentation of a phenomenon, where characteristics are measured in accidental time steps with different instruments and its mathematical model.

---

[5] Kohonen, T., **Self-Organization and Associative Memory**, Second Edition, Springer-Verlag, Berlin, 1988.

[6] Grabec, I**., Self-Organization of Neurons Described by the Maximum-Entropy Principle**, Biol. Cybern., **63**, pp. 403-409, 1990.

The noise can appear also in cases where we want to valuate a phenomenon in whichever way possible, for example a political situation in the country - the problem is the influence of the subjectivity of different observers. Obviously all real-life problems involve the dealing with noise.

The noise depends, besides on the errors which can result from a measurement, also on the way of modeling a phenomenon. This is the case, when the description of a phenomenon considers a (too) small number of measurements, or a (too) large number of measurements with small errors. Therefore, at least a principal knowledge about a phenomenon is of great importance. But when we treat a new phenomenon this is often impossible. Solving this dichotomy involves the use of different statistical tools (e.g. parametric analysis), which make it possible to determine the significance and/or the influence of a single parameter. Unfortunately, such tools are usually very sophisticated and divert users from the application. In comparison to those tools, the aiNet allows simplified, physically clear estimation of the noise in the data and assessment of the importance of single parameters. More about this topic you will find in Chapter 3.

1.4 Selecting the Parameters of the Phenomenon - General

Let us go back to the phenomenon of the day-temperature changing and assume we have a tool (e.g. the aiNet), which is able to describe the phenomenon perfectly. A used mathematical model demands only two parameters for the description (hour of the day and the temperature at that time). Everything works fine until instead of the mapping (see Figure 1.5)

$$day\_hour \rightarrow temperature\ ,$$

one demands an inverse mapping. It is obviously, that we have difficulty while the function, which defines the model, is not one-to-one. How to solve the problem?

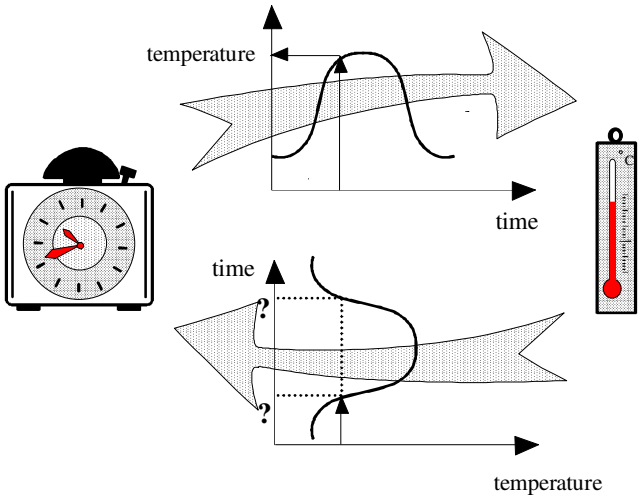

Figure 1.5: Mathematical relations in modeling the phenomenon of the day-temperature changing.

We will have a look just at the simplest solution. Exact mathematical mapping

$$temperature \rightarrow day\_hour$$

has a practical meaning. To get the right answer, we need to ask the right question first, which means the right modeling of the phenomenon in such cases. The new, additional parameter must be introduced for this reason (see Chapter 2). It will describe an ascendant and a descendent part of the curve. In other words: morning and afternoon, if the highest temperature is reached at noon. A new description of the mathematical model has new dimension and the solutions, given by that model, lie on two parallel plains (see Figure 1.6). As can be seen, additional parameter gives a simple information, the solution lying on corresponding plain.

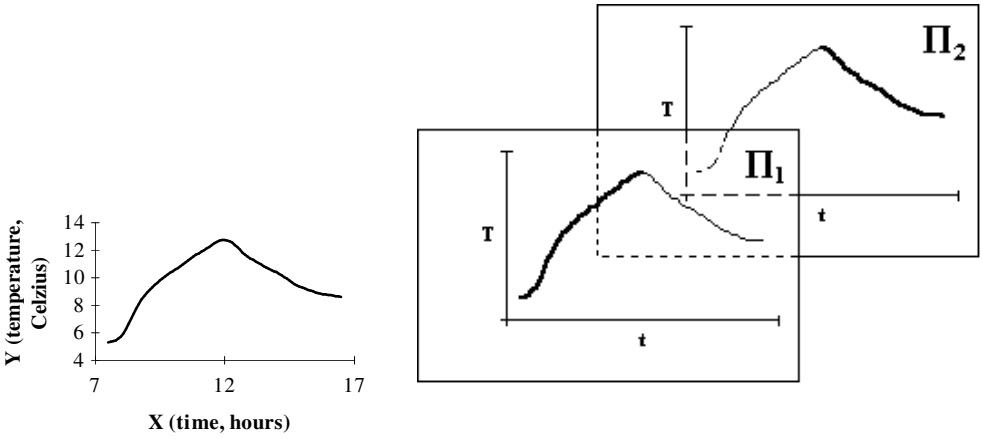

Figure 1.6: Graphical presentation of the mathematical model of the day-temperature changing in case of additional parameter.

Efficient preparation of the model demands logical parameter selection; the best choice is the case, where parameter has practical, physical meaning. The above example shows the introduction of additional parameter due to the inverse reversion, which divides the description of the phenomenon into two parts.

1.5 Efficiency of the Model

We already analyzed the problem with a small number of data in the beginning of the chapter. Somehow it is clear that the reliability of the solution, and further the efficiency of the model strongly depend on the quality of the data base (reliability of the measurements and/or collected facts), and representativity of the samples. Representativity of the samples means an approximately uniform distribution of the samples in the problem space and, of course, sufficient number of those samples.

The above sentences are illustrated on a simple example. Figure 1.7 shows the results of measurements and two solutions which can be obtained in cases of different reliability levels, based on the measured data.

We might want to have a model, which gives exact or almost exact results in comparison with the given data. A general solution of such model is shown in Figure 1.7./b/. One recognizes a very rough curve, which seems unnatural, but goes through the measured points (And this is what we demand!). According to the fact, we do not know much about the phenomenon (small number of measurements of already unreliable source), it should be nonsensical to demand a great precision of the model. So we tolerate greater deviations in solutions for the available data. The

consequence of such modeling shows Figure 1.7./c/. The solution curve seems much more natural than before. Moreover, such model has very good generalization capabilities.

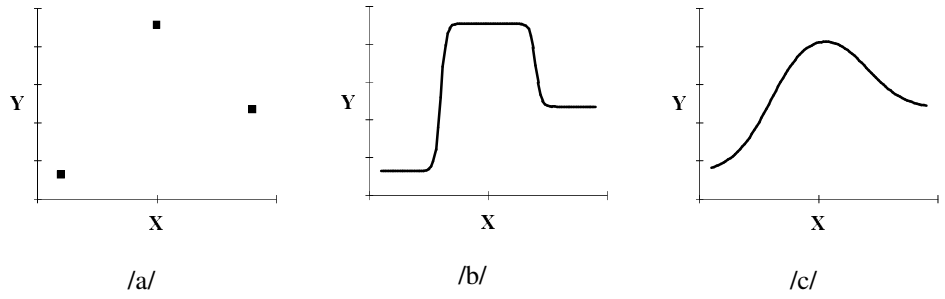

Figure 1.7: Two models of the phenomenon of the day-temperature changing for different reliability levels.

Principally, we should not demand a too high precision (exaggerating precision!), neither a too low precision (superficity!). How to choose the right precision then? The aiNet is provided with tools, which allow determination of the optimal solution for the available dataset. The problem is reduced to the classical optimization problem, which, for example, can be solved in the iteration process by a procedure, called a bisection (see Figure 1.8).

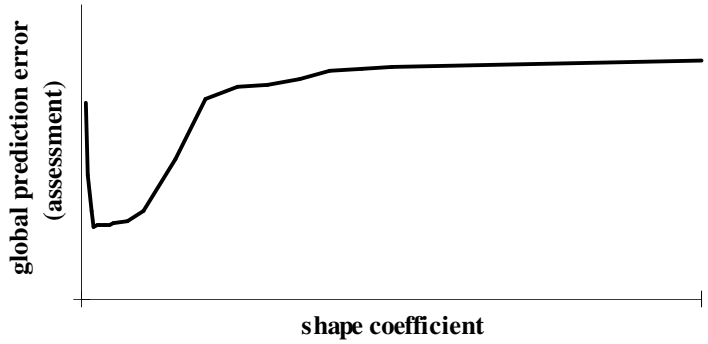

Figure 1.8: Minimum (prediction) error for the available dataset.

It should be mentioned that the reliability of the model for the available dataset is regulated with one parameter only. We will name it shape coefficient for the time being, while, as can be seen from the Figure 1.7, it influences on the shape of the solution curve (solution hyperplane).

1.6 Brief Review of Basic Concepts

Let us see, what we understand under the concepts *modeling* and *model*.

Modeling of the phenomenon means the selection of the *right attributes* of a phenomenon and a suitable coding of the *right number of measurements and/or facts* in ***the mathematical form***. The

aiNet program is provided with tools, which allow successful modeling in a defined sense, and they will be presented in the chapters.

A model is a dataset or a data base itself. Data base consists of model vectors (see next chapter) which describe the phenomenon. Its components represent the values of the parameters - variables of the phenomena. We speak about a kind of a *non-parametric model*, where the model is not presented in the explicit form.

Chapter 2

2. Parameter Types and Preparation of Model Vectors

2.1 Basic concepts

Basic concepts will be explained through the example of the phenomenon of the day-temperature changing again. Temperature was measured each hour; measurement was started at 8:00 in the morning, and ended at 4:00 p.m. We can describe the phenomenon with a group of 9 vectors, so called ***model vectors***:

model vector	**=**	**time**	**temp.**
mv^1	=	8.00	5.30
mv^2	=	9.00	9.10
mv^3	=	10.00	10.50
mv^4	=	11.00	11.70
mv^5	=	12.00	13.10
mv^6	=	13.00	11.30
mv^7	=	14.00	10.40
mv^8	=	15.00	9.20
mv^9	=	16.00	8.70

Figure 2.1: Model vectors for the description of the phenomenon of the day-temperature changing.

Each model vector has two components - phenomenon is described with two ***parameters***: time, when measurement was done, and the temperature at that time. General formulation of model vector can be written as:

$$\{\mathbf{mv}\}^T = \{param\_1,\ \ param\_2\}^T = \{time, temperature\}^T .$$

And finally, each phenomena can be written in general form as follows:

model vector	=	param_1	param_2	...	param_M
mv_1	=	$value_{11}$	$value_{12}$	...	$value_{1M}$
mv_2	=	$value_{21}$	$value_{22}$	...	$value_{2N}$
.	.	.	.		.
.	.	.	.		.
mv_N	=	$value_{N1}$	$value_{N2}$	...	$value_{NM}$

Figure 2.2: General description of the phenomenon in mathematical form.

Defining what we want to know about the phenomenon it is necessary to introduce the concept of ***input parameter*** and ***output parameter***. For example, when we want to know *what temperature* there was *at an appointed hour* the input parameter is the time (hour) of measurement, and the output parameter is the temperature at that time (hour).

Generally, input and output parameters are complementary. If we imagine that input parameters are written as one partial vector and output vector as another partial vector, concatenation of both vectors gives a new vector (model vector, as we note), which completely describes the phenomenon. The model of the phenomenon, consisting of model vectors with known values of input and output parameters, is able to predict unknown values of output parameters of so called prediction vectors. Naturally, the values of input parameters of prediction vectors are known.

Let us take an example of a phenomenon, which is completely described with N model vectors. Each model vector (**mv**) has M input parameters and one output parameter (M+1 st parameter of the model vector). We want to use the model for the prediction of missing values - values of output parameter of the same phenomena for three new prediction vectors (**pv**), which have known values of input parameters only.

model vector	=	input param_1	input param_2	...	input param_M	output param_1
$\mathbf{mv_1}$	=	m_value_{11}	m_value_{12}	...	m_value_{1M}	$m\_value_{1,M+1}$
$\mathbf{mv_2}$	=	m_value_{21}	m_value_{22}	...	m_value_{2M}	$m\_value_{2,M+1}$
.	.	.	.		.	.
.	.	.	.		.	.
$\mathbf{mv_N}$	=	m_value_{N1}	m_value_{N2}	...	m_value_{NM}	$m\_value_{N,M+1}$
$\mathbf{pv_1}$	=	$m\_value_{N+1,1}$	$m\_value_{N+1,2}$	...	$m\_value_{N+1,M}$	$p\_value_{1,M+1}$
$\mathbf{pv_2}$	=	$m\_value_{N+2,1}$	$m\_value_{N+2,2}$	...	$m\_value_{N+2,M}$	$p\_value_{2,M+1}$
$\mathbf{pv_3}$	=	$m\_value_{N+3,1}$	$m\_value_{N+3,2}$	...	$m\_value_{N+3,M}$	$p\_value_{3,M+1}$

Figure 2.3: Mathematical description of the phenomenon in case of prediction of unknown values of output parameters.

model vector	=	time	temperature
mv_1	=	8.00	5.30
mv_2	=	9.00	9.10
mv_3	=	10.00	10.50
mv_4	=	11.00	11.70
mv_5	=	12.00	13.10
mv_6	=	13.00	11.30
mv_7	=	14.00	10.40
mv_8	=	15.00	9.20
mv_9	=	16.00	8.70

pv_1	=	10.50	11.15
pv_2	=	11.25	12.75
pv_3	=	15.75	8.95

Figure 2.4: Predicted values of the temperature for the selected time points.

Common scheme from Figure 2.3 has for the case from the Figure 2.1 the form as can be seen from Figure 2.4 (we assume, that our interest are the unknown temperatures at 10^{30} a.m., at 11^{15} a.m. and at 15^{45} p.m.). Predicted values are written in shadowed boxes. It can be seen from the example, that the model predicts the unknown values through the interpolation and/or the extrapolation. In the procedure it uses optional a non-linear function, whose shape is defined by the penalty coefficient (see Chapter 3).

2.2 Selecting parameter types

We can always choose between more options in the modeling process. Basically, we distinguish between two types of parameters:

- uniform type ($-\infty$, $+\infty$) and
- discrete type (integers only, e.g. 1, 2, 3, ...).

Discrete type of parameter is just a special case of uniform type. It was introduced with the intention to speed up the computing time, and for the sake of the simplifying the modeling in some cases. Its written form is more comprehensive and more convenient for the use in some special cases. Unfortunately, discrete type of parameter can not be used as output parameter! Nevertheless, if we want to predict the value, described already by the discrete parameter, it is necessary to change the description of the phenomenon. How to do this, will be shown in the following subsections.

There is one important fact to be mentioned: the use of different statistical methods for the description of phenomena is very pretentious in cases of mixing different parameter types, and is tied to many limitations. Neural networks (e.g. program aiNet) and neural network-like systems do not demand any special procedures and do not introduce any restrictions in such cases.

Let us see the use of both parameter types in different cases. We will take the phenomenon of the day-temperature changing again. If we measure the time and the temperature at that time, the most logical selection seems to be the ***uniform parameter type*** for both variables of the phenomenon. Variables can have theoretically all the values between 0 and 24 (if the time is

measured in hours) or between -273°C and $+\infty$ °C (if the temperature is measured in degrees of Celsius). Model vector has a very simple form and can be written as is shown in Figure 2.1.

The phenomenon is non-linear. To allow inverse mapping, new additional parameter must be introduced. Part of the curve, either ascendant or descendant, can be defined with a new ***discrete parameter type*** (see Figure 1.5, Chapter 1). Model vectors can be written as follows:

model vector	=	**time**	**temperature**	**period**
mv_1	=	8.00	5.30	1
mv_2	=	9.00	9.10	1
mv_3	=	10.00	10.50	1
mv_4	=	11.00	11.70	1
mv_5	=	12.00	13.10	1
mv_6	=	13.00	11.30	2
mv_7	=	14.00	10.40	2
mv_8	=	15.00	9.20	2
mv_9	=	16.00	8.70	2

Figure 2.5: Written form of the phenomenon of the day-temperature changing in case of the third discrete parameter type.

All model vectors are transformed to the new, abstract hyperspace before the computation (prediction; see *Normalize Model Vectors* in User's Manual, Part 1!). For this reason, the value size of the discrete parameter type has no influence on the final prediction results or modeling of the phenomenon. For the sake of simplicity, which is important for the user, it is recommended to use integer numbers in ascendant order (e.g. 1 and 2 in above example from Figure 2.5).

2.3 Preparation of model vectors in some other cases

Let us take a look at an interesting case, when in the evening the temperature suddenly increases due to the temperature inversion (see Figure 2.6). We get the second ascendant part of the curve with the same sign of the derivate. Formally the same coding can be used for the complete description of the phenomenon as in the previous case, only the third part of the curve has its own discrete value.

model vector	=	time	temperature	period
mv_1	=	8.00	5.30	1
mv_2	=	9.00	9.10	1
mv_3	=	10.00	10.50	1
mv_4	=	11.00	11.70	1
mv_5	=	12.00	13.10	1
mv_6	=	13.00	11.30	2
mv_7	=	14.00	10.40	2
mv_8	=	15.00	9.20	2
mv_9	=	16.00	8.70	2
mv_{10}	=	17.00	9.30	3
mv_{11}	=	18.00	9.70	3

Figure 2.6: Coding of the phenomenon in case of suddenly increased afternoon temperature.

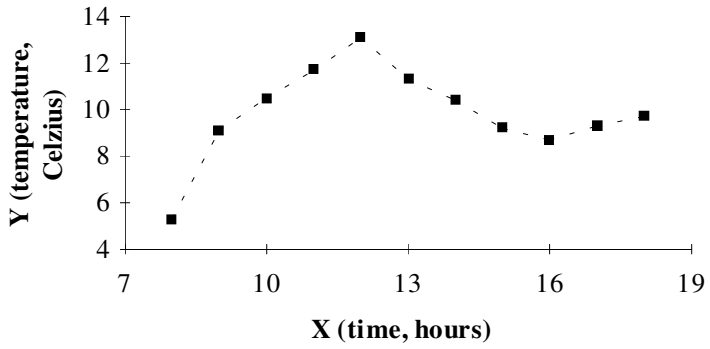

Figure 2.7. Graphical presentation of the phenomenon in case of suddenly increased afternoon temperature.

Generally, the description of the phenomena is independent of the selection of input and output parameters (what is logical). For example, the model based on the description from Figure 2.6 simply gives the answers to the questions; what was the temperature at certain hour in the morning, in the afternoon or in the evening.

And what to do, when we know the hour, the temperature at that hour, and we are interesting in the answer to the trivial question, when did it happen? While the discrete parameter type was used, the answer can not be obtained at once, as mentioned earlier. But we can simulate discrete parameter type with the uniform type as follows:

model vector	=	time	temperature	morning	afternoon	evening
mv_1	=	8.00	5.30	1.0	0.0	0.0
mv_2	=	9.00	9.10	1.0	0.0	0.0
mv_3	=	10.00	10.50	1.0	0.0	0.0
mv_4	=	11.00	11.70	1.0	0.0	0.0
mv_5	=	12.00	13.10	1.0	0.0	0.0
mv_6	=	13.00	11.30	0.0	1.0	0.0
mv_7	=	14.00	10.40	0.0	1.0	0.0
mv_8	=	15.00	9.20	0.0	1.0	0.0
mv_9	=	16.00	8.70	0.0	1.0	0.0
mv_{10}	=	17.00	9.30	0.0	0.0	1.0
mv_{11}	=	18.00	9.70	0.0	0.0	1.0
pv_1	=	10.50	11.15	0.99	0.01	0.00
pv_2	=	11.25	12.75	0.99	0.01	0.00
pv_3	=	15.75	8.95	0.01	0.99	0.00

Figure 2.8: Description of the phenomenon of the day-temperature changing in case of coding it with the uniform parameter types only, and prediction for three cases.

The answer to the question above will give us different values for each output variable of the phenomenon, belonging to output parameters of the uniform type. These values can be interpreted as the certainties of each event. These events in above cases are certainties, that the temperature at the appointed hour was reached in the morning (first output parameter), in the afternoon (second output parameter), and in the evening (third output parameter).

The final solution in Figure 2.8 is the event with the highest value. Results show that the model was successfully predicting all time periods. The right answers have values higher than 99%.

It should be mentioned that both descriptions are identical in the case when time period arises as input parameter. The computation is faster, when the description with the discrete types of parameters is used.

2.4 Conclusion

Descriptions of basic concepts introduced in Chapter 2 will be given once again.

Model vectors represent a way of coding the phenomenon in the mathematical form as vectors whose components are important variables of the phenomena. Variables of the phenomenon are called *parameters* of the phenomenon, which are further divided *into input parameters* and *output parameters*. Basically, we differ between two types: *a uniform type* and *a discrete type*.

There follow some of our general advice.

- During the preparation of the knowledge base (or data base) the assurance of representativity of model vectors is of great importance. It is well known that neural networks give very good results in cases of interpolation in problem space, but possibly very poor, or even useless in cases of extrapolation. We need to assure, that model vectors "cover" all, or at least as many combinations of extreme values (minimum and maximum values) of input parameters as possible. In the case where all combinations of input parameters of model vectors are possible, the necessary number of model vectors with extreme values is at least 2^N. N

represents the number of input parameters in a model vector (input variables of the phenomenon).

- In some practical examples, when the number of input parameters is large, above demand might be to excessive. Luckily, practical experience shows, that such examples are very rare. Although this might happen we will model the phenomenon with large number of variables and the above demand will not be satisfied. Results might be useful or even good, but we must bear in mind, that some of predictions were obtained by extrapolation, which is physically questionable.
- Preparation of the efficient model depends on the appropriate selection of variables of the phenomenon on one side, and the sufficient number of descriptors of the phenomenon (model vectors) on the other. In practice we are usually limited with the number of model vectors. Between the number of variables of the phenomenon (number of parameters) and the number of model vectors there is a relation, which can not be determined analytically. It is commonly valid, that a higher number of variables of phenomenon demands a higher number of model vectors for sufficient modeling.

Chapter 3

3. Measures for the Estimation of Prediction Error and Preparation of Efficient Model

3.1 General

The use of classical neural networks (e.g. back propagation neural network - BP NN) introduces the problem of training (learning). Instead of the concentration on the problem itself, a lot of energy must be put into the determination of the learning coefficients. Here is the point where science comes very close to the art while, except in rare cases, there are no general valid rules or mathematical expressions, which define procedures or ways to selection of learning coefficients so far. Literature, and experience also, shows, that the selection of learning coefficients in many cases strongly depends on the concrete phenomenon. The selection of learning coefficients in the best case can be generalized to very similar phenomena. Here is also one unavoidable, usually a very unpleasant problem - the noise in the data. The user (researcher) never knows, if the neural network is stuck in some local minima, if the data are very noisy and if not even a better result is possible, and so on.

The aiNet program, compared to the other similar tools for dealing with neural networks, allows creative work. The researcher can concentrate most of her/his time to the modeling of the phenomenon. For this reason, she/he has at a disposal a few, very simple tools to use. We will not describe the theoretical backgrounds in detail here; it should be mentioned only, that all the methods are based on the multidimensional non-parametric regression (see [7,],[8]).

Taking into account the assumption, that the preparation of model vectors is independent of the tool, we want to use, it can be find out very quickly, that the final aiNet model depends on one coefficient only (see subsection *Penalty coefficient*). With regard to the BP NN, which demands the selection of at least three learning coefficients, selection the number of hidden layers and the number of neurons in these hidden layers, the use of the aiNet becomes evident. While the aiNet does not demand classical learning in most of the practical cases, the order of presentation of model vectors is not important, as well (commonly, we can choose among at least four

---

[7] Grabec, I. & Sachse, W., **Automatic Modeling of Physical Phenomena: Application to Ultrasonic Data**, J. Appl. Phys., 69 (9), 1991.

[8] Ramirez, M., R. & Arghya, D., **A faster learning algorithm for back-propagation neural networks in NDE applications**, Proceedings of the 2nd International Conference on AI, pp. 275-283, 1991.

presentations, see [9]). The used procedure is not sensitive to the noise in the data. Therefore, the model can give us good results even in cases of very high noise level. The final judgment of validation of the above statements in various cases, and comparison of the efficiency of various neural network models (the aiNet models and the models, prepared by other neural network programs) is left to the users of the program.

3.2 Measures for the Estimation of Prediction Error

Modeling phenomena urgently demands mathematical assessments of the prediction error or reliability of the results*. The definition of the global or local prediction error is independent of the used method (filtration, verification, prediction; see the explanation in next subsections). It can be simply written as:

$$E = x_m - x_p,$$

where lower indexes *m* and *p* present the measured value of the parameter and the predicted value of the parameter, respectively. In a more complicated description

$$E_{ij} = x_{m\_ij} - x_{p\_ij}$$

the lower index *i* describes the i-th model vector and lower index *j* the j-th parameter of the model vector. Such defined error is called ***signed local error***.

Signed local error has sense in cases only, where there is no noise in the data or it is very small. In all other cases we determine ***absolute local error***. As the name tells, the description is similar to the description of signed local error. Local error is determined as the absolute value:

$$\overline{E}_{ij} = \left| x_{m\_ij} - x_{p\_ij} \right|$$

RMS (prediction) error on the parameter is then defined as square root of sum squares (error power square) divided by the number of model vectors (N):

$$G_j = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(E_{ij}\right)^2}$$

Total RMS (prediction) error is defined similarly, the expression under the square root is divided with the number of all output parameters (M), as well:

$$G = \sqrt{\frac{1}{N * M}\sum_{j=1}^{M}\sum_{i=1}^{N}\left(E_{ij}\right)^2}$$

---

[9] Ramirez, M., R. & Arghya, D., **A faster learning algorithm for back-propagation neural networks in NDE applications**, Proceedings of the 2nd International Conference on AI, pp. 275-283, 1991.

* The aiNet gives assessment of the prediction error on the whole data base; the procedure is described in details in belonging subsections!

It should be mentioned that when there is only one output parameter, the RMS prediction error on the parameter is the same as the total RMS prediction error.

Two different options can be used for normalizing model vectors in the aiNet. The RMS error on parameter and total RMS error are therefore not directly compared with similarly defined measures in other tools, which use the transformation in the unit hyperspace.

3.3 Penalty coefficient

Different statistical methods demand the selection of the shape (type) of the function, which suits the best to the description of the phenomenon. The coefficients of empirical (regression) equations are determined then with the method of the least squares. In such a way we try to describe the phenomenon with some in advanced presumed empirical law. While the data are usually incomplete, the selected law is fitted very good to the available data and usually fails when more data is acquired. The available data bases are not representative in most of the practical cases and therefore the automatic modeling of the phenomenon is very appropriate when new data are obtained.

Compared to the above mentioned parametric methods, the non-parametric[**] model, prepared by the aiNet, does not need any in advanced presumed law. Also, when the data are changed, or new data are added to the data base, the aiNet adapts the model automatically. Only one coefficient, so called ***penalty coefficient***, must be defined. It has an indirect relation to the learning error (and/or learning threshold) in classical neural networks - both influence the final solution. The influence of the penalty coefficient on the solution will be shown on the concrete example in the third part of the User's Manual.

Penalty coefficient determines the shape of the curve in two-dimensional problems, and the shape of the hyperplane in more dimensional problems, respectively. While penalty coefficient influences the accuracy and/or the efficiency of the model, the determination of its optimal value represents the key point in a modeling process. A few tools which help the users find the right value of penalty coefficient and help them better understand the phenomenon are described in the next subsections. It should be mentioned at this point, that there are few possibilities for changing penalty coefficient in the problem space. The most simple variant is the one, where penalty coefficient has a constant value. The other possibilities are mostly dynamical adaptations, where its value is additionally modified for each point in the hyperspace by different built-in methods.

3.4 Modeling tools

Modeling tools will be explained on the example of the day-temperature changing again. The temperature at the beginning, and at the end of the experiment was measured in two, relatively short time steps with the same instrument. Somewhere in the middle of the experiment the temperature was measured in two accidental time points with two different instruments (see Figure 3.1).

[**] The name *parameter* in this particular case relates to the coefficients of regression equations which are usually used for the description of phenomena!

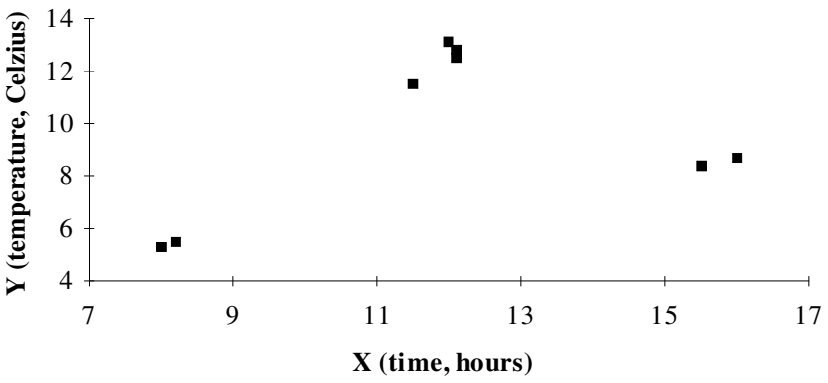

Figure 3.1. Phenomenon of the day-temperature changing - example of one of many possible descriptions.

Filtration tool

The most basic tool in the modeling process is the filtration. It gives the most straight forward estimation of the noise in the data. Value of *each output parameter for each model vector* from the data base is predicted on the basis of *all model vectors* from the data base. Selected small value of penalty coefficient allows the estimation of the noise in the data with one of the RMS global errors (choice depends on the user and its knowledge about the phenomenon).

It should be noted, that the noise is not only the function of errors in measurements, or function of subjectivity in preparation of model vectors. It might happen that one of the important influencing parameters is excluded (is not taken into account) from the modeling process. Even theoretically perfect data base will give in such cases the error in the prediction which can not be avoided. The error is equal to the influence of the excluded parameter. High noise (estimation of one of the RMS errors) might be a good indicator of exclusion of one or more influencing parameters of the phenomenon. The problem is not entirely trivial, while the phenomenon might be chaotic, too.

Very noisy data might significantly slow down or even block the training of usual neural networks in some cases. Filtration is a very useful tool is these cases. It can be used as a filter for the preparation of the new data base, which is free (or almost free) of noise. The data can be normally used then for the training of other neural networks, for example, BP NN.

Verification tool

Verification of models in case of using classical neural networks is based on a very simple principle: two thirds of the available data is used for the preparation of the model, and the last third is used for the verification (the selection of model vectors in both groups is accidental!). A basic problem in such procedures is the size of available data bases in reality - in most cases we have only a limited number of model vectors. If from already limited number of model vectors one third has been cut, a very truncated data base is left. Model, prepared on the truncated data base can give poor predictions, which further leads to the conclusion that the model is useless.

On the contrary, verification in the aiNet program works on a very simple principle. The value of *each output parameter for each model vector* from the data base is predicted on the basis of *all* ***other*** *model vectors* from the data base, which means that the model vector under consideration is

temporarily removed from the data base. Such procedure gives the global error estimation which is a ***better*** global error estimation, while it is determined on the whole data base. We might say, that this is the prediction error. The last statement is more correct the more representative the available data base, and the higher the number of model vectors.

Prediction tool

The key tool in the sense of the use of efficient non-parametric model is prediction. When the model is prepared (with both above described tools, selected all important input parameters of the phenomenon and selected the value of penalty coefficient), prediction can be used for determination of output parameters of the phenomenon for model vectors with only known input parameters. We want to predict the unknown values of the variables, which completely describe the phenomenon in one, special case, for example:

- we want to know the temperature at 01:32 p.m.in case of the day-temperature changing,
- what is the diagnosis in case of back pain, if the patient is up to 60 years, and the pain appears just on one side of his back, and so on.

Prediction is a tool, intended for the end user.

3.5 Relations between modeling tools

Qualitative relations between the measured and the predicted values, which are given by different modeling tools, are shown in Figure 3.2. The case from Figure 3.1 is used as a test example!

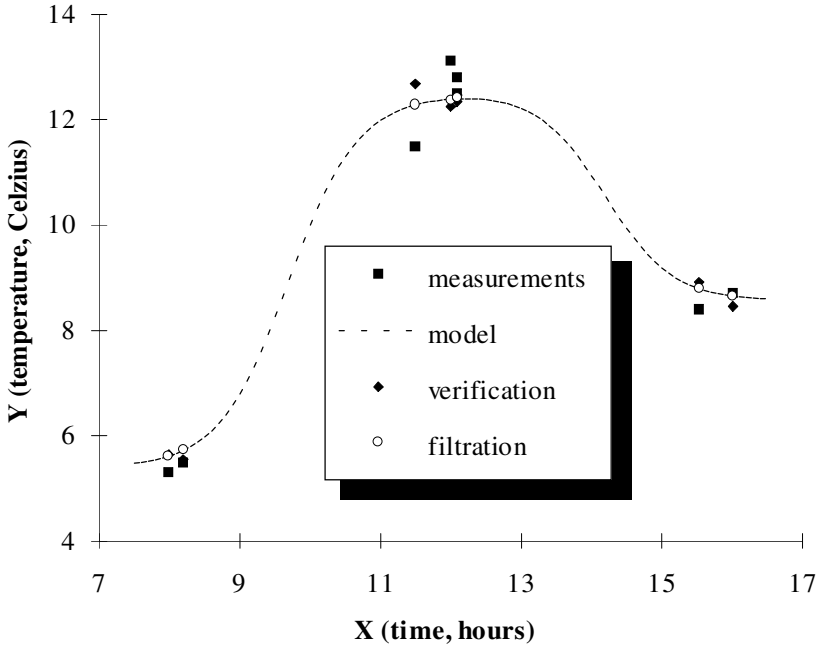

Figure 3.2. Relations between measured and predicted values for different modeling tools.

Solid squares show the measured data. Dashed line shows the curve, which describes the phenomenon on the basis of all measurements (model vectors from the data base), and it has been calculated with a prediction tool. A filtration tool determines the points on that curve (labeled with a circle). Verification determines points (labeled with solid diamond); in general, these points have the greatest distances to the measured points. The Figure shows also that the filtration tool gives the lowest value for the global error estimation, and the verification tool the highest value. In the case of a perfect data base and a very large number of model vectors both estimations converge in the single value which, in reality, will almost never happened .

Figure 3.2 shows another interesting fact. We have more data in an area (approximately at noon), which seems to be noisy. Both absolute local errors, obtained either by a filtration tool or by a verification tool are approximately the same in some points in this area. This does not only prove the above statement, but it also shows that the aiNet works without difficulties with noisy data.

The illustration of the influence of the penalty parameter on the solution will be shown just quantitatively. We chose the case with static (constant) value of the penalty coefficient in the problem domain. Figure 3.3 shows qualitative relations between estimations of global errors for both modeling tools. The curve of verification tool has a shape known from the optimization problems. Optimal solution is obtained at one single value of penalty coefficient. This is not the optimal solution, but it is the optimal solution at the available data base. As can be seen from Figure 3.3, the optimal value of penalty coefficient can be simply determined by different methods, e.g. bisection.

Error estimation of filtration tool always converges in the some value. If this value is zero and the data base is representative, we can state that the noise in the data is very low. The higher is the value, the more noisy the data and the less efficient the model.

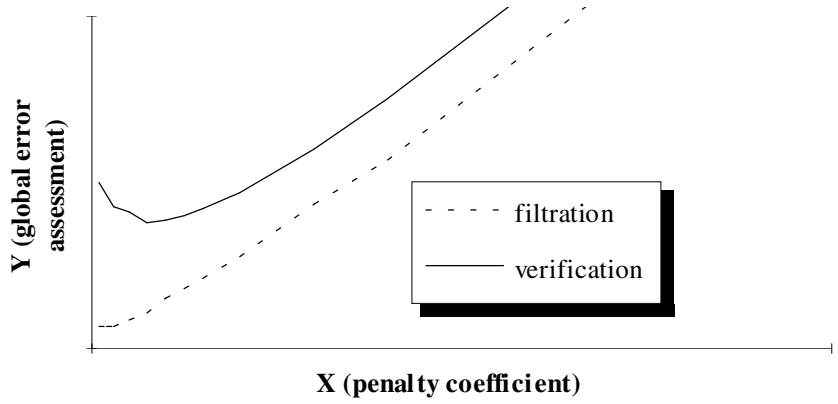

Figure 3.3. Relations between global error estimations for filtration and verification for variation of penalty coefficient.

3.6 Assistant tools

The aiNet program contains some assistant tools which allow simple control and visualization of predicted results. These are different graphical tools which show comparison between the predicted and the measured results of the phenomenon, and the estimations of absolute local errors for model vectors from the data base. Assistant tools can be used in combination with all other modeling tools (filtration, verification, ...).

For the end user the estimation of either signed or absolute prediction local error is of the great importance. For this reason the prediction tool gives the estimation of local error, too. While it strongly depends on local characteristics of the phenomenon, we suggest the use of local error estimation in prediction together with one of global error estimations, obtained with the verification or the filtration tool. Experienced users (We hope, all of you will become experienced users!) can prepare their own criteria for the reliability of the model, based on these two error estimations.

Figure 3.4 shows the estimation of absolute local error for the case from Figure 3.1. The example is changed in a way that a higher noise was added to some measured data in the middle area (approximately at noon), and some additional measured data in equidistant time points were added.

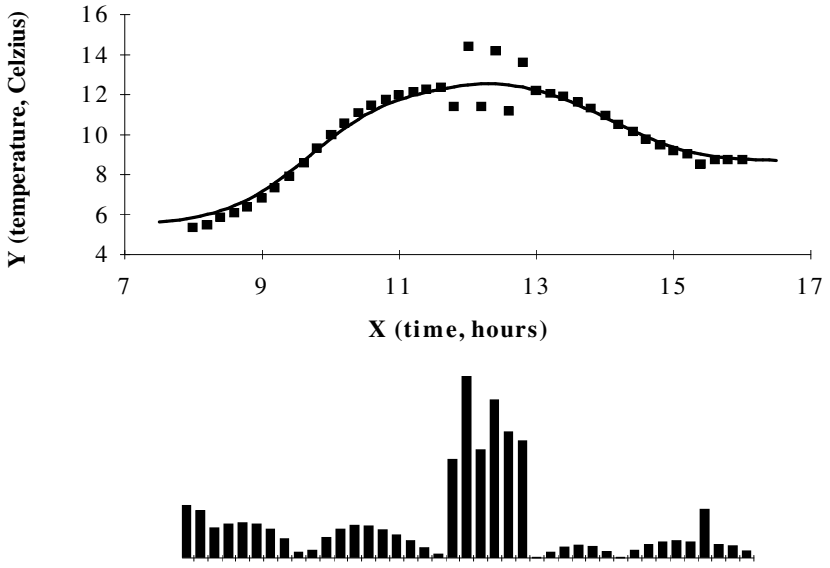

Figure 3.4. Estimation of absolute local error.

New, mostly graphical tools are in a preparation and testing phase now. Among the others we would like to mention the graphical tool, which will be build in the aiNet in the future.

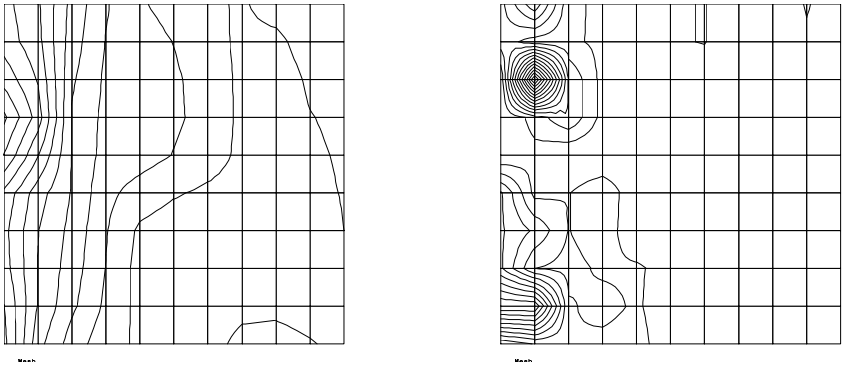

Figure 3.5: Graphical presentations of prediction, and distribution of local prediction error.

It will show the isolines of equal values of the output parameter at variation of two input parameters and fixed or excluded values of other input parameters. One of important results might be a distribution of the estimation of local prediction error, which can be shown in the same manner in the same or in another picture (see Figure 3.5). A theoretical basis of these tools along with main ideas will be presented when tools have been included into the program.

Chapter 4

4. *Conclusion

This part of User's Manual briefly describes basic concepts about modeling with the aiNet, and the available tools in the computer program. The authors expect that the brief description will offer appropriate base for work with the program. Additional assistant tools will be prepared in the future. They will simplify the modeling process and the aiNet will become even more user friendly.

Authors gratefully acknowledge all comments and warnings, either in the user's manual or in the computer program. Each idea will be welcome and will help to a greater efficiency of the program in the future.

Chapter 5

5. References

1. Ramirez, M., R. & Arghya, D., **A faster learning algorithm for back-propagation neural networks in NDE applications**, Proceedings of the 2nd International Conference on AI, pp. 275-283, 1991.

2. Samaad, T., **Backpropagation Improvements based on Heuristic Arguments**, Theory Track, Neural and Cognitive Sciences Track, International Joint Conference on Neural Networks, Vol. 1, Washington, D.C., 1990.

3. Johansson, E., M., Dowla, F., U. & Goodman, D., M**., Backpropagation learning for Multi-layer Feed-forward Neural Nets Using the Conjugate Gradient Method**, Lawrence Livermore National Laboratory, 1990.

4. Shanno, D., F., **Conjugate Gradient Methods with Inexact Searches**, Mathematics of Operations Research, Vol. 3, pp. 244-256, 1978.

5. Kohonen, T., **Self-Organization and Associative Memory**, Second Edition, Springer-Verlag, Berlin, 1988.

6. Grabec, I**., Self-Organization of Neurons Described by the Maximum-Entropy Principle**, Biol. Cybern., **63**, pp. 403-409, 1990.

7. Grabec, I. & Sachse, W., **Automatic Modeling of Physical Phenomena: Application to Ultrasonic Data**, J. Appl. Phys., 69 (9), 1991.

8. Grabec, I., **Modeling of Natural Phenomena by a Self-Organizing System**, Proc. ECPD NEUROCOMPUTING, Vol. 1, No. 1, 1990.

9. Casciati, F. & Peruš, I., **Neural Networks: an Alternative Answer to Structural Engineering Needs**, ECC Worskhop on ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS TRAINING IN STRUCTURES, Athens, January, 1994.

Part 3:
Examples

Introduction

This part of manual includes practical examples from many different science areas. Every example begins with a general introduction of the problem and the presentation of the goal. This is followed by the modeling section, where we explain how we model the phenomenon, which tools were used and what we expect to obtain. The model should be verified, and for this reason we explain which value of penalty coefficient was used, how we obtained it, and why we used that particular value. Comments give some additional information and important notes, which might help users in modeling their specific phenomenon. Because the presented examples are shown at different levels some of these subsections are not strictly the same as we described above. We hope it will not bother you too much.

The authors wish to thank a few of them who were suspicious at the beginning of their use with neural networks, and to few of them almost always dissatisfied with the efficiency of any kind of neural networks we wanted to prove they were wrong and here is the aiNet.
We sincerely hope you will like it.

Iztok Perus

Example 1

1. GENERAL PROBLEM: Time series

1.1 General

One of the central tasks of the empirical science is to build models on the basis of experimental data. Practical example of such problem is the modeling of a time series. We can recognize very quickly, that the modeling of a time series is not a typical technical problem only, but is also a characteristic of many other real life disciplines, e.g. stock- and gold-market prediction. A recent research in a chaotic time series showed that neural networks exceed conventional linear and polynomial predictive methods by many orders of magnitude. We will not bother you here with a chaotic time series; we will give a simple illustrative sinusoidal example. It will be shown, that very simple example can help us to understand the phenomenon and help us to avoid many difficulties, which might appear in more complicated cases.

1.2 Modeling of the Phenomenon

A time series can be described mathematically as

$$Y = f(X), \text{ where } X=X_0, X_1, X_2, ..., X_N$$

or in vector form as

$$\{Y\}^T = \{Y_0, Y_1, Y_2, ..., Y_N\}^T$$

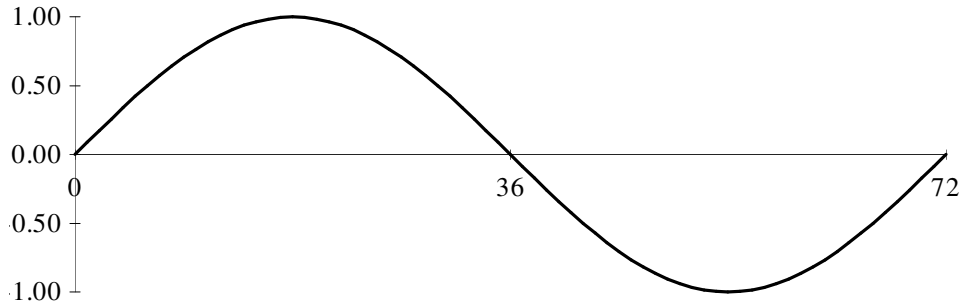

Figure 1.1. Sinusoidal function, described with 72 model vectors.

The variables depends on time. The description seems more natural, if we write

$$\{Y(t)\}^T = \{Y(t_0), Y(t_1), Y(t_2), ..., Y(t_N)\}^T$$

Let us define the problem now. The three sequential sinusoidal values are used as the input data to predict the fourth, the output one. The same description is used as many times as necessary to completely describe one sinusoidal period. Each model vector can be written in general form as:

$$\{\mathbf{mv}\}^T = \{Y(t_1), Y(t_2), Y(t_3); Y(t_4)\}^T$$

1.3 Verification of the model

The first step in a verification of the model was determination of the optimal value of penalty coefficient by the verification tool. After several trials the value 0.003 was found as the optimal value. Then we started with a prediction, where in each sequential step previously predicted values were used. This procedure can be written in general form as:

1. step: feed sin(t-2), sin(t-1), sin(t-0) to the network to compute sin(t+1),
2. step: feed sin(t-1), sin(t-0), sin(t+1) to the network to compute sin(t+2),
3. step: feed sin(t-0), sin(t+1), sin(t+2) to the network to compute sin(t+3),
4. step: continue with the same procedure for as many steps you want!

Figure 1.2 shows the results of the prediction for three periods. As can be seen, the results are somehow unnatural - the model in not able to predict descending near the top (maximum values) and climbing near the bottom (minimum values) of the curve.

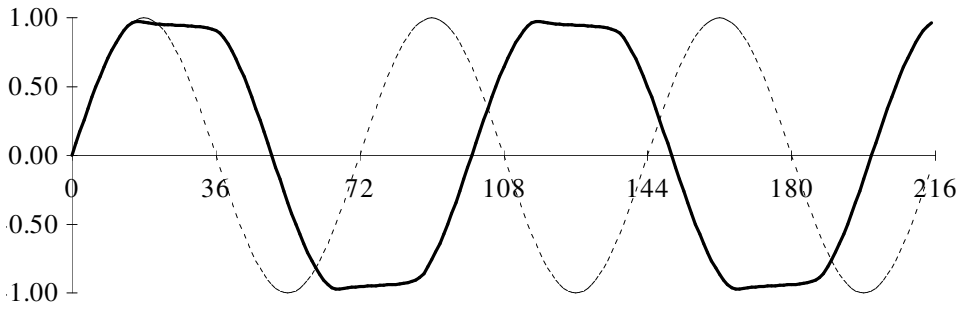

Figure 1.2: Prediction of sinusoidal time series at penalty coefficient = 0.003.

While the solution was not what we wanted, we repeated the prediction again, this time with a lower value of penalty coefficient. The results, which were better, are shown in the Figure 1.3.

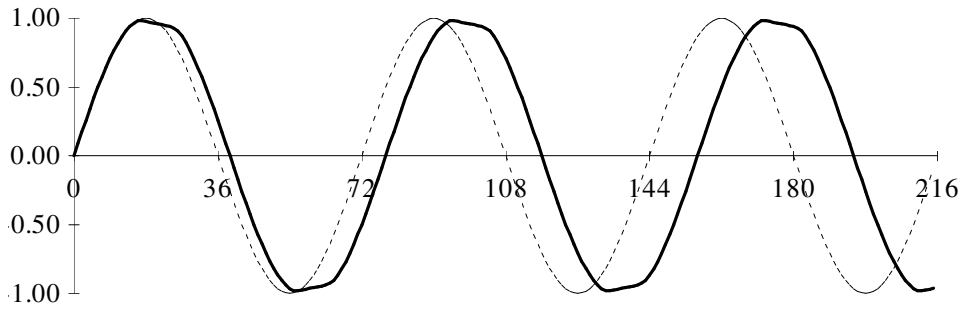

Figure 1.3: Prediction of sinusoidal time series at penalty coefficient = 0.002.

The results from Figure 1.3 suggest a very low value of penalty coefficient. After the prediction has been repeated the third time with penalty coefficient = 0.001, the predicted values were almost exact. We obtained a true sinusoidal curve, except very near the top and the bottom of the curve again; there appears a slight differences, which normally can not be seen in the graphic presentation.

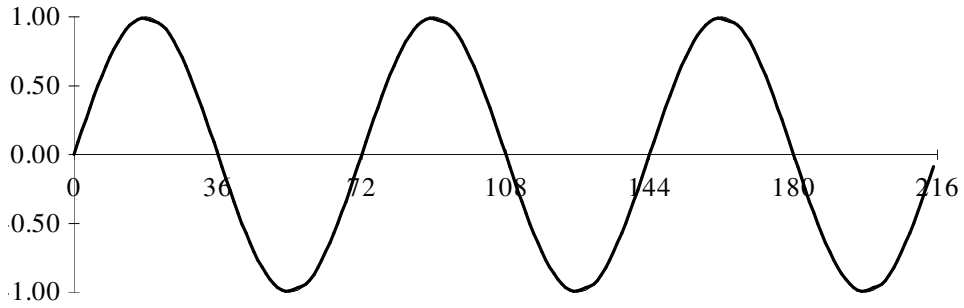

Figure 1.4: Prediction of sinusoidal time series at very low penalty coefficient (0.001).

Let us see the results in the case of a higher penalty coefficient, for example: at 0.01. This situation, which can be seen as a result of many classical neural networks, is shown in Figure 1.5. As it was mentioned in the User's Manual, Part 2, penalty coefficient is strongly correlated with the learning error in the BP NN. A greater allowed learning error (learning threshold) corresponds to the a higher penalty coefficient. The situation where the learning threshold is set too high the BP NN can not learn the phenomena sufficiently. A time series is very sensitive to disturbances in the data, so the BP NN should be trained perfectly for the acceptable predictions. And this corresponds to a very low penalty coefficient.

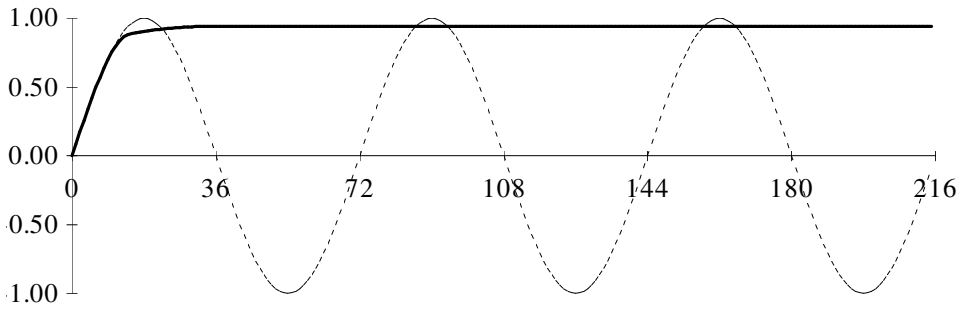

Figure 1.5: Prediction of sinusoidal time series at penalty coefficient = 0.01.

NOTE: Input data (learning set) are written in the data base, named EXAMPLE1.CSV! The predicted results, shown in Figures 1.2, 1.3, and 1.4 are written in the data base EXAMPLE1.RES. Users can check these values by simulating the above described procedure by typing manually the predicted values in the aiNet prediction window.

1.4 Comments

Users can try to use the example with 360 discrete points. They will find out that more data will not solve the problem. Only very high precision (small value of penalty coefficient) will give the right results. This points out two very important facts:

- the problem is very sensitive to small disturbances (prediction errors!); prediction of time series might show unusual behavior in more complicated cases.
- a lot of data can not guarantee a good prediction. For example, let us take weather forecasting. If we should be able to take into account every molecule around the earth, we can predict (theoretically) the weather exactly. But such huge amount of data demands very high precision, which can be lost through the computation process. Therefore it can be concluded, that there exists not only the optimal value of penalty coefficient, but also the optimal ratio between variables of the phenomenon, and an acceptable prediction, according to the available (software and hardware) tools.

Example 2

2. GENERAL PROBLEM: Approximation of Non-linear Functions

2.1 General

Neural networks and neural network-like systems are approximators already by their nature [10]. For this reason they can be used for the approximation of non-linear functions. The use of the aiNet program will be shown on simple example of the day-temperature changing serving as a basic example in the explanation of concepts in User's Manual, Part 2. The influence of penalty coefficient is shown first. This is followed by the selection of parameters (model vector preparation) in different situations.

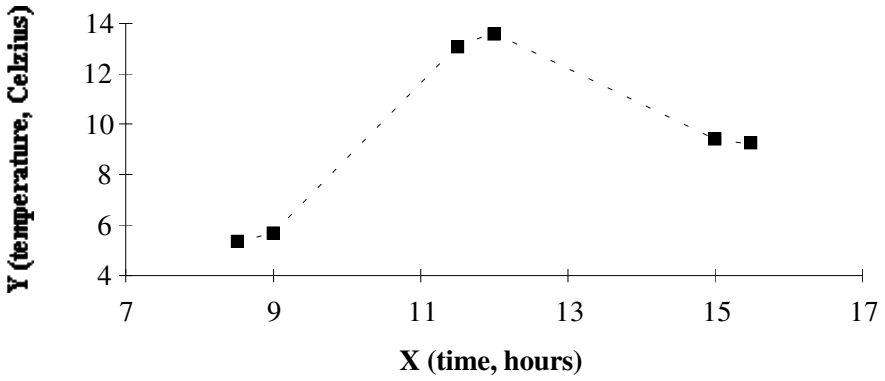

Figure 2.1: The phenomenon of the day-temperature changing, described by six measurements.

---

[10] Carpenter, W., C. and Barthelemy, J., F., **Common misconceptions about neural networks as approximators**, Proceeding of the 3rd International Conference on the Application of Artificial Intelligence to Civil and Structural Engineering, in Neural Networks and Combinatorial Optimization in Civil and Structural Engineering (Topping, B. H. V., editor), pp. 11-18, 1993.

The phenomenon of the day-temperature changing is described by six measurements. Mathematically, this can be presented by six model vectors. Time points are shown in horizontal direction (usually labeled as X), while day temperatures are shown in vertical direction (usually labeled as Y).

2.2 Modeling of the Phenomenon

We are interested in the day temperature as a function of time, which, mathematically appears as:

$$T = T(t)$$

Figure 2.2 shows solutions of the above equation. Results are obtained by prediction tool (see the aiNet program: *Prediction*) in discrete time points, starting at 7:30 by step length of 6 minutes. It should be noted that the aiNet gives different solutions depending on the value of penalty coefficient.

Figure 2.2./a/ clearly shows that a very low value of penalty coefficient gives a very rough curve. In comparison to the classical BP neural network this condition corresponds to a state, where neural network memorizes learning patterns very well (model vectors from the data base, according to our convention). A generalization of such learned neural networks is usually inadequate, while networks learn model vectors very good.

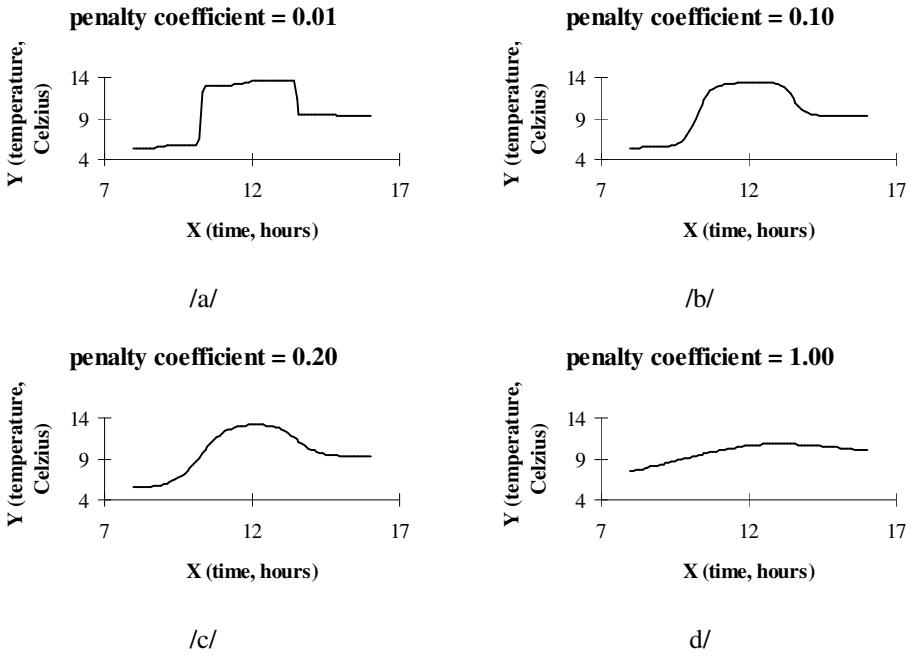

Figure 2.2: Graphical presentation of modeling the day-temperature changing phenomenon as a function of penalty coefficient.

Opposite to a low value is a high value (relative) of penalty coefficient (see Figure 2.2./d/). Large values of penalty coefficient give a very smooth curve, furthermore, increasing the value of penalty coefficient leads to a constant value.

Figures 2.2./b/ and 2.2./c/ show two intermediate states. It is evident that the best solution is obtained by using the penalty coefficient value of 0.20 (see Figure 2.2./c/).

Suitability (validation) of the aiNet model can be assessed visually. This is limited to very simple cases. In multidimensional cases, which are the cases in real life, suitability can be assessed on the basis of various measures, given in the aiNet program (see User's Manual, Part 2, Chapter 3). The assessment of *RMS error on parameter* or/and *total RMS error* is usually used. In the above example we tried to determine the penalty coefficient value according to the total RMS error. It should be noted, that a verification tool is used for this purpose.

Now, we look at the inverse problem. The above example gives the solution of an equation T = T(t). How to model the phenomenon, when we want to obtain the inverse mapping.

$$t = t(T)$$

For example, we want to know, when some appointed temperature was reached? While the function, which describes the phenomena, is not one-to-one, we have to deal with the problem as it is described in the User's Manual, Part 2 (see Chapter 1 and Chapter 2). Model vectors should be instead of

model vector	=	**time**	**temperature**
mv_1	=	8.50	5.30
mv_2	=	9.00	5.70
mv_3	=	11.50	13.10
mv_4	=	12.00	13.60
mv_5	=	15.00	9.40
mv_6	=	15.50	9.20

written as

model vector	=	**time**	**temperature**	**period**
mv_1	=	8.50	5.30	1
mv_2	=	9.00	5.70	1
mv_3	=	11.50	13.10	1
mv_4	=	12.00	13.60	2
mv_5	=	15.00	9.40	2
mv_6	=	15.50	9.20	2

The third discrete parameter describes the ascending part of the curve (morning) when it has value "1", and descending part of the curve (afternoon) when it has value "2". In the mathematical sense, the description of the phenomenon is not changed at all.

There are now two input parameters: temperature and period (part of the day), when the temperature is the object of interest (morning [1], afternoon [2]). An output parameter is the time point in which the temperature has been given value.

NOTE: There are two data bases. The first one EXAMPL2A.CSV includes an example, when the phenomenon is described with only two parameters (one input and one output). The second EXAMPL2B.CSV includes an example, when the phenomenon is described with three parameters (two inputs and one output). The whole phenomenon is described by six model vectors. Results from Figure 2.2 are obtained on the basis of input data, written in the EXAMPL2A.PRD file.

2.3 Verification of the Model

While the shown example is of illustrative nature, the model is not verified by available mathematical tools, provided in the aiNet. However, we can conclude very quickly that the model gives visually acceptable results at value 0.20 for penalty coefficient (see Figure 2.2./c/). The answers to the given questions in case of using the third (discrete) parameter, are exact at the same value of penalty coefficient, too.

2.4 Comments

If the mathematical description of the phenomenon is not using the third (discrete) parameter, the results will represent just average values at times when an appointed temperature is reached. Such results do not have any practical value or applicable meaning. We need to be careful in modeling the phenomenon, which should be described in such a way, that inverse mapping, if they are necessary, can be obtained without problems.

To the question, when was the temperature 8^{o}C reached in the morning, the aiNet produces an answer: at 8 o'clock and 55 minutes. The value of penalty coefficient was 0.15. One of interesting thing that should be mentioned is that the aiNet model can tell us also, when the morning or afternoon is. In this case the time is the input parameter.

Example 3

3. SENSOR PROCESSING: Character Recognition

3.1 General

Unlike mathematics and science, which, respectively, pursue pure beauty and an understanding of nature, the (neurocomputing) technology pursues the development of useful things. From the earliest days of neurocomputing, neural networks have been recognized as being exceptionally well suited for solving the problems in sensor processing. While neural networks can handle a large amount of information at once, they have opened doors to interesting applications in the area of pattern recognition - the process of visually interpreting and classifying symbols. It should be mentioned that sensor processing primarily means pattern recognition, although other functions such as filtering and data compression are also important application areas. [11]

The problem of an automatically recognizing characters in printed or hand-written material has been studied for decades (for example, see [12] and [13]). A number of technical and market successes have been achieved in character recognition. United States Postal Service uses character recognizers extensively for reading printed addresses on letters, banks utilize character recognition machines for reading the numerical amounts of checks etc.

In character recognition there are two important performance measures: *acceptance rate* and *substitution rate*. The acceptance rate is a number of characters per million (in per cent) that the system accepts as readable. The substitution rate is a number of characters per million (in per cent) accepted characters that the system classifies incorrectly. Both of these systems rates are measured by testing the system on huge volumes of text drawn at random from the environment in which the system is expected to operate. Best systems achieved an acceptance rate of approximately 95%, and a substitution rate of less than 5%.

---

[11] Hecht-Nielsen, R., **Neurocomputing**, Addison-Wesley Publishing Company, Inc., 1990.

[12] Guyon, L. et al, **Comparing different neural network architectures for classifying handwriting digits**, Proc. of the Int. Joint Conf. on Neural Networks, **II**, pp. 127-132, IEEE Press, New York, June 1989.

[13] Hecht-Nielsen, R., **Nearest matched filter classification of spatiotemporal patterns**, Applied Optics, **26** (10), pp. 1892-1899, 1987.

A simple example of character recognition will be shown here, primarily to illustrate how to use the aiNet program in such cases. The problem can be defined as a classification of single numerals written by hand. A pixel feature description, without pre-processing is used in a present example, although no serious researcher suggests such a description as being appropriate for character recognition. We will show two different ways of coding the pixel information and the obtained results in both cases.

3.2 Modeling

Signs were constructed in a box which is divided into 49 equal sub-boxes: each box is divided into 7 equal-width columns and 7 equal-width rows, as shown in Figure 3.1. The recognizing system should be able to distinguish three different signs, which represent numbers 0, 1 and 2. Figure 3.1 shows the numbers presented with 12 different symbols (four samples of each number). Accordingly, data base consists of 12 model vectors.

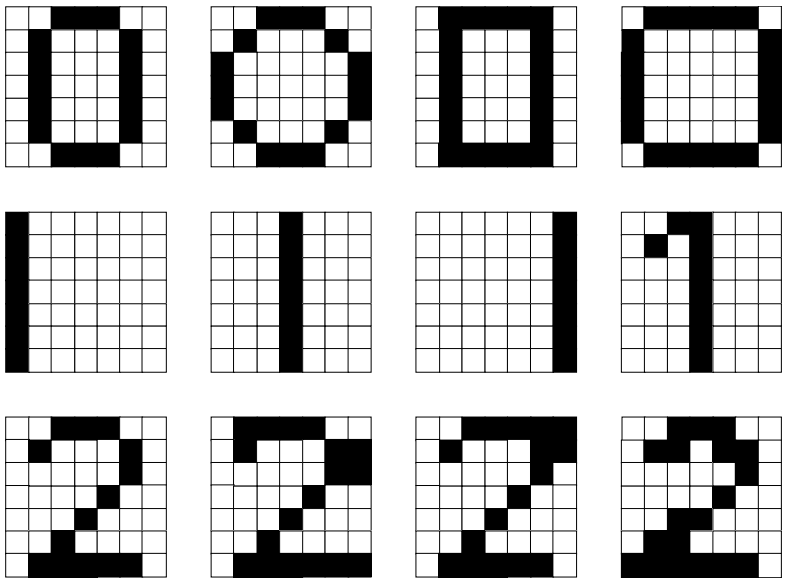

Figure 3.1: Symbols for numbers 0, 1 and 2.

Firstly, we indicate where the character line goes through each of the 49 sub-boxes (or better, when the line "cover" more than 50% area of the sub-box). If the numeral line passes through the i-th sub-box, the corresponding value in the model vector is set to "1"; otherwise the corresponding value is left set to "0".

As mentioned before, model vectors can be prepared in two different ways. In the first case the model vector is constructed of columns, which columns are being indexed from the left to the right. Such model vector has 49 input parameters (a 7 x 7 mesh!) and three output parameters (three different symbols!), all together 52 parameters. In the second case each input parameter represents a sum of the black pixels in the directions of both main diagonals of the box. Determined in such a way, the model vector has 26 input parameters (13 sums in one direction + 13 sums in the other direction!) and three output parameters (again three different

symbols!), all together 29 parameters. A Simple example from Figure 3.2 shows both ways of coding model vectors.

$\{\mathbf{mv}\}^T = \{1, 0, 0, 1, 1, 1, 0, 0, 0\}^T$

$\{\mathbf{mv}\}^T = \{1, 1, 1, 1, 0, 0, 1, 2, 1, 0\}^T$

Figure 3.2: Two ways of coding model vectors in case of recognizing characters.

Preparation of the model vector in the second case shows the possibility of a simple reduction of the amount of necessary information. The number of input parameters in the first case decreases from 49 to 26 in the second case.

NOTE: There are two data bases. First data base EXAMPL3A.CSV contains model vectors, which have 52 parameters (the first way of coding model vectors). The second one EXAMPL3B.CSV contains model vectors, where the symbols are described with 29 parameters (the second way of coding model vectors). The whole phenomenon (recognition between three different symbols) is described with 12 model vectors. Results in both cases are obtained using two files, EXAMPL3A.PRD and EXAMPL3B.PRD.

3.3 Verification of the Model

Efficiency of both models was established at different values of penalty parameter. Figure 3.3 shows test examples - these are four noisy numbers (one "1", two "2" and one "0", respectively).

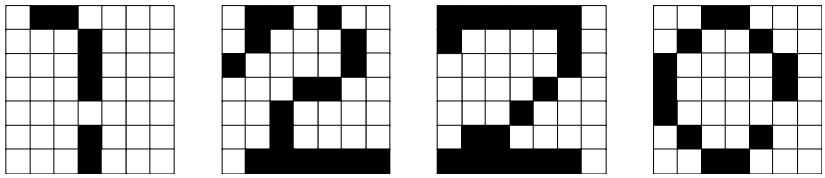

Figure 3.3: Noisy numbers 1, 2, 2 and 0.

We can find out very quickly, that both aiNet models recognize all numbers without difficulties. The right predictions have extremely high values (certainties) in case of low values for penalty coefficient. Such a result is logical, and is expected, while noisy numbers are still very similar to the given numbers in the data base.

High values for penalty coefficient give lower certainties for the right predictions, but still the highest in comparison to the other possible solutions. Even a very high value for penalty coefficient (e.g. 100.0, see Table 3.1) gives the greatest value for the right prediction in both models. The result shows an interesting fact, that the number of necessary information can be reduced even more to achieve applicable results in the given example. This can have a

practical and useful meaning in cases where we deal with a huge number of symbols and large vectors, which describe them.

Table 3.1: Results for both models at different values for penalty coefficient.

penalty coefficient	model 1				model 2			
	No. 1	No. 2	No. 2	No. 0	No. 1	No. 2	No. 2	No. 0
1.0	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.95
5.0	1.00	1.00	1.00	0.96	0.99	0.99	0.95	0.66
5.0	1.00	1.00	1.00	0.82	0.92	0.73	0.82	0.55
50.0	0.56	0.53	0.54	0.37	0.40	0.37	0.39	0.36
100.0	0.44	0.43	0.44	0.35	0.37	0.35	0.36	0.35

3.4 Comments

A present example is of more illustrative nature. It shows the simplest variants only; how to prepare model vectors and the influence penalty coefficient to final a solution. We can say without exaggerating that the aiNet can be used for solving real problems in the area of pattern recognition, as a recognizing system of either printed or hand-written characters. It can also be used for other problems in sensor processing, as data compression and noise removal from time-series signals.

Example 4

4. ECONOMY: Regional Analysis - Regional Development Model

4.1 General

In view of development characteristics in a modern world countries have been divided into a number of areas - regions, which, due to their specific characteristics such as smallness and a better overview over the whole area, are a much more suitable area to encourage development than a country as a whole. In almost all countries there is a trend towards a decentralization of decision-making in politics as well as in economy. Besides its branch component the development policy of a country also has a regional component, which makes the overall policy of a country even more complicated. An economic and a political optimum can be imposed for a whole country but this optimum collapses with the introduction of regional optimums. All those optimums should be taken into account but they can, however, cause a competition and disharmony among other regional interests and even disagreement with a state. It is therefore evident, that the question of (under)development of a region plays a very important role in the development policy of a country.

One of the initial problems in a regional analysis is a regionalization e.i. dividing a particular area (country) into regions. Economists have no unified criterion - some do not define a region at all, some define its specific characteristics and some provide a concrete definition. We will not deal with this problem here, we shall suppose that its solution is already known. It is our purpose to develop a model on basis of which and with the help of the most important parameters we will be able to distinguish between a developed and underdeveloped region and the level of its (under)development.

4.2 Modeling

When dealing with a problem mathematically, economists are usually unable to exactly tell us why they came to a certain conclusion. On the other hand, they can easily provide conclusions in some well known cases (e.g. a region is undeveloped, if its GNP is lower than 1), which represent their knowledge about the problem. Therefore, we first drew up a questionnaire in view of five most important indicators of regional development. The questionnaire has been filled out by several experts. On the basis of the filled questionnaires we prepared a data base, which abridged though not less successful, serves as a test example. There are more techniques to be implemented in the preparation of the data base and the end model. They shall be included in the program in the next versions.

The most important indicators signifying the level of regional development (*LRD*) are[+] :

- general national product per capita - *GNP*
- a public sector investment share in a state investment (%) [*I_invest*]
- population growth index (number of new-born babies in one year / number of deaths in one year * 100) [*I_pop*]
- a portion of students at colleges and universities in the population, expressed by indices (number of students at colleges and universities / number of all inhabitants * 100) [*I_stud*]
- a portion of employees compared in the population, expressed by indices (number of employees/ number of all inhabitants * 100) [*I_employe*]

A model vector can in general be expressed as follows:

$$\{\text{mv}\}^T = \{GNP, I\_invest, I\_pop, I\_stud, I\_employe; LRD\}^T$$

NOTE: There is only one data base for the above example - EXAMPLE4.CSV. The phenomenon has been described by 243 model vectors, each of them is presented by six of the above parameters.

4.3 Verification of the Model

In the model of regional development we take an interest in the development compared with other regions, therefore, the greatness of penalty coefficient is of no vital importance. We should only pay attention to the fact that the solution is smooth enough, as is generally the case in the nature. In using the model we suggest a penalty coefficient between 0.15 and 0.25. We can even choose a higher value but to the extent that the model will still not give us a constant value as a result.

4.4 Comments

The results obtained by the model can be regarded in two ways:

- we simply want to know if a region is developed (LRD > 5.5.) or underdeveloped (LRD < 5.5.) and /or
- we are interested in the LRD compared with the other regions. The LRD is important in case of different measures imposed by a state to encourage development in those regions

This model has some advantages over the usual statistical methods or over diagrams with quite simplified criteria. Let us mention the most important ones:

- we can take into account as many indicators as we know and can sufficiently evaluate

---

[+] The question of indicators presents a special problem and is not dealt with here specifically. The point is, we want to show how successful a method can be, however, a number and importance of indicators playing no vital role in it.

- the development can be observed by different combinations of indicators
- the model can be relatively quickly adjusted to new experience, new indicators can be added or the old ones which proved unuseful excluded
- reversed relations can be obtained very easily : e.g. we can find out parts of investments needed to make a region a developed one (or leave it undeveloped) if other indicators are known (fixed).

Figure 4.1 shows in which way the LRD depends on two indicators: the *GNP* and investments. It should be mentioned that the helping tool, which enables the depiction of Figure 4.1 will be available in the next version of the aiNet program.

It is evident that the model should offer the known trends: the greater the *GNP* the higher the regional development, the greater the share of investments, the higher the regional development. It is also obvious that the *GNP* is a much more important indicator than investments. Correspondence of the results with the known quality relations only proves the usefulness of the model, which, on the other hand, enables the quantity evaluation of regional development.

Figure 4.1: Dependence of regional development on two most important indicators: GNP (horizontal) and investments (vertical)

Example 5

5. CIVIL ENGINEERING: Diagnosis of Damage of Prestressed Concrete Piles During Driving

5.1 General

In driving prestressed concrete piles (PCP), cracking and spalling may be encountered. The damage to such piles can be classified into three major types:

- spalling of concrete at the head of the pile due to high compressive stress,
- spalling of concrete at the point of the pile due to hard driving resistance at that point, and
- transverse cracking or breaking of the pile due to the combination of torsion and reflected tensile stress.

The damage depends on many factors, like subsoil conditions, the features of the pile composition and inadequate or improper methods and equipment. While damage results in time delays, injuries and cost overruns, diagnosis of the causes of damage is very important.

Visual damages such as cracks and concrete spalls at different locations, together with the known characteristics of PCP indicate the causes of those damages. The judgment of causes is strongly based on human experts' knowledge. Usually, a human being needs years of study or practice to achieve the status of an experienced expert.

Human expert solutions involve two problems: dependency on the expert and possible human subjectivity. The best way to avoid these problems is to formalize the expert knowledge from different sources and represent it in the mathematical form. In formalizing the expert knowledge, one encounters some major problems, including knowledge representation, reasoning and acquisition. Among those problems, knowledge acquisition might represents the most difficult task in diagnosis. Usually, human experts can not explain, how they reason and why do they obtain such conclusions. Procedural solution can therefore not be used. The use of artificial neural networks seems to be one of good alternative solutions. The expert knowledge can be represented in form of samples (rules) acquired from human experts. Such samples consist of two parts: the first part represents features - descriptions of visual damages and other known characteristics of PCP during driving, the second part represents the causes of damages.

5.2 Modeling

The knowledge base, as named the database in this case, was obtained directly from a paper published by Yeh et all [14]. 18 different visual damages and some known characteristics of PCP during driving were considered as follows:

[1] spalling of concrete occurred at the head of the pile
[2] spalling of concrete occurred below the head of the pile
[3] cracking occurred on one side of the pile
[4] cracking occurred around the pile
[5] cracking occurred at pile slit
[6] direction of cracking is longitudinal
[7] direction of cracking is transverse
[8] the slenderness of the pile is too high
[9] pile has splices
[10] driven times of cushion are many
[11] cushion is broken after pile-head breaking
[12] cushion is crushed after pile-head breaking is broken
[13] spiral reinforcement after pile-head breaking is broken
[14] ratio of driving formula resistance to design resistance is exceeded
[15] penetration of pile driving has ever increased suddenly
[16] penetration of pile driving has ever decreased suddenly
[17] pile hole after driving is full with soil and water
[18] pile hole after driving is full with concrete spall

Many different causes are possible during driving PCP. The most probable causes, which the model can propose as a solution, are as follows:

[1] uniformity of cushion material is poor
[2] ductility of cushion material is poor
[3] cushion is overused
[4] energy of hammer is to large
[5] driving in very hard rock or soil
[6] driving in very soft soil or cave
[7] concrete honeycomb
[8] strength of concrete is insufficient
[9] strength of pile splices is insufficient
[10] strength of pile tip is insufficient
[11] strength of prestress in insufficient
[12] strength of spiral reinforcement is insufficient

A uniform type of the parameter was used to describe the phenomenon: each visual damage or known PCP characteristic is labeled as "0" or as "1"; the same convention is used in labeling possible causes. "1" means certainty of visual damage, certainty of known PCP characteristics or certainty of possible cause, and vice versa for "0". For example, if cracking occurred on one

---

[14] Yeh, Y., Kuo, Y. & Hsu, D**., Building KBES for diagnosis PC pile with artificial neural network**, Journal of Computing in Civil Engineering, Vol. 7, No. 1, pp. 71 - 93, 1993.

side of the pile, the third component of the model vector has value "1", or "0" otherwise. Finally, each sample in the knowledge base is presented as a model vector, which has components either zeros or ones.

Model vector

$$\{\mathbf{mv}\}^T = (0,1,1,0,0,1,0,1,0,0,1,0,0,0,0,0,0,0;\ 1,0,0,0,0,0,0,0,0,0,0,0)^T$$

represents the situation, *where spalling of concrete occurred below the head of the pile*, *cracking occurred on one side of the pile*, *direction of cracking is longitudinal, the slenderness of the pile is too high*, and *the cushion is broken after pile-head breaking*. A possible cause is that *the uniformity of cushion material is poor*.

NOTE: Database or knowledge base in this case, is written in the EXAMPLE5.CSV file. The phenomenon is completely described by 240 model vectors. Each model vector has 30 parameters, where 18 of them are input parameters and the rest (12) are output parameters.

5.3 Verification of the Model

Usually, available data bases for certain problems do not contain representative samples. Non-representative samples do not "cover" all possible cases - all possible combinations of input parameters. An optimal solution, based on such data is therefore not the actual optimal solution, yet the optimal solution on instantaneously available data base. However, the penalty coefficient was determined through iteration process. The optimal prediction error was defined as the one, where the number of wrong prediction is minimal.

The analysis of the problem shows that very small values of penalty coefficient give always only one cause for each sample from the data base (which is a reasonable result). The value of output parameter ascribed to this cause is almost equal to one. On the other hand, higher values of penalty coefficient give values greater than zero for almost all output parameters - possible causes. These values can be interpreted as certainties for each cause.

An "exact" solution is defined as one, where the right cause has the highest value among all possible causes. It showed up that the best value of penalty coefficient should be between 0.20 in 0.50.

The results of analysis are shown first as histograms of prediction errors, which are combinations of two presentations. Ground-plan of histograms shows correlation (or comparison!) between predicted and actual results, while axonometric view shows the number of right predictions for all possible causes.(see Figure 5.1). Dependency of the optimal solution on the penalty coefficient is shown in Figure 5.2.

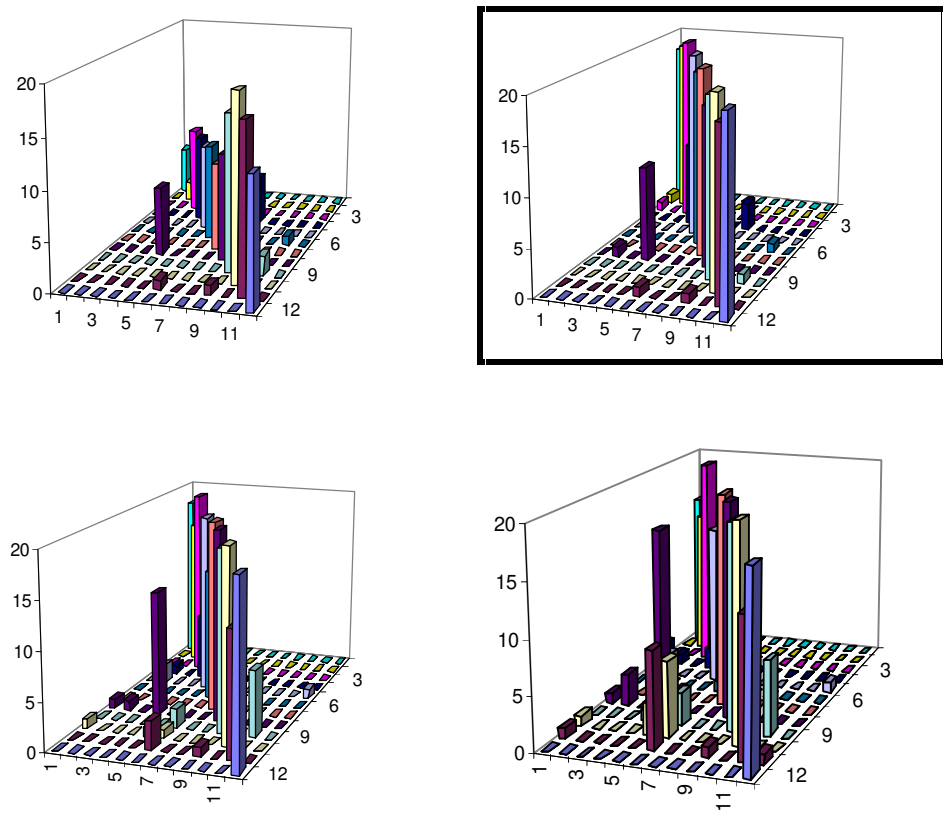

Figure 5.1. Histograms of prediction errors at different values for penalty coefficient.

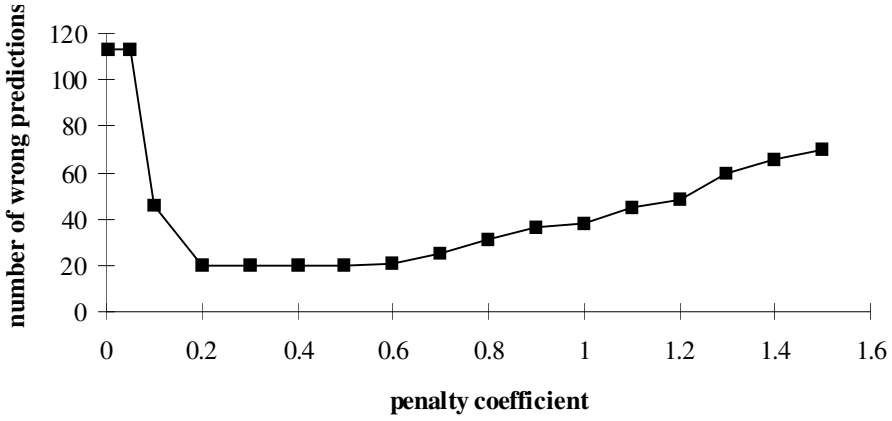

Figure 5.2. Dependency of the optimal solution on the penalty coefficient.

Figure 5.2 shows the same number of exact predictions for penalty coefficient values between 0.2 and 0.5. While the prediction is better there is less disagreement between predicted and actual results and the value (certainty) is higher for actual cause in case of wrong prediction, the optimal solution is obtained at penalty coefficient 0.5 (see Figure 5.1./b/). We propose this value in case, when the aiNet model will be used in practice for diagnosis of damage of PCP during driving.

NOTE: User's version of the aiNet uses two different types of normalization: *regular* and *statistical*. In case of regular normalization model vectors are automatically transformed linearly from real world to abstract hyperspace with dimensions (-1, 1) for each parameter. Nevertheless, the results of this example are obtained on the basis of transformation to abstract hyperspace with dimensions (0, 1) for each parameter. The use of different hyperspaces does not influence the solution significantly, only the optimal value of penalty coefficient is different!

5.4 Comments

The use of neural networks and/or neural network-like intelligent systems shows practical experiences and very good results in the field of civil engineering, too. The aiNet program in this case was used for diagnosis of damage of PCP during driving. The example represents a special case of qualitative reasoning, which differ from quantitative reasoning predominantly in the way of coding the data base (or knowledge base). We can conclude, that technical applications demand an appropriate tolerance level, which assure the optimality of the solution. Generally, we must not be too precise (large number of digits in results of analysis). It should be noted that precision of the model and optimal solution depend not only on penalty coefficient, but also on the size and quality of the data base. In connection with human logical reasoning this means the result of reasoning is more certain the more and better information about the phenomenon we have.

Example 6

6. MEDICINE: Diagnosis in Case of Back Pain

6.1 General

Most of the people have pain in the back from time to time. The doctors usually do not even try to determine the nature of a disease from observation of such symptoms, while pain mostly disappears after short repose. The doctors indicate such pain as an unspecified back pain. This illness is the main reason of lost working days all over the world [15]. All those people, who carry or lift up heavy objects are prone to a back pain. Not only they, but also the others who sit a lot in the same position or those who have to be crooked in an unnatural position, have problems with their back. In any case, the back pain might be a sign of many diseases. The right diagnose in case of back pain (as is the case with all symptoms) is of extraordinary importance for the health of the patient (on time medical treatment!).

The given example which shows diagnosis of the newly occurred back pain, mostly serves as an illustration for the use of the aiNet program. In such cases one usually decides to make a use of expert system; we will show how to use the aiNet as representative of neural networks in this case.

It should be mentioned that expert systems have one advantage compared to neural networks, which might be of great importance: a relatively simple procedure allows explanation of all steps in the reasoning process (consultation!). Neural networks have, on the other hand, capability of generalization, which might also be of great importance in most cases of diagnosis: solutions can be obtained also for the samples which are not explicitly codified in the knowledge base.

6.2 Modeling

Most of natural science problems, especially problems which can be measured, have simple relations between their quantities. These relations are of the same kind and are described mathematically by operators (only "***and***", only "***or***", only "***and/or***"). Such operators are captured automatically in the knowledge base. On the other hand, a diagnosis operates with

---

[15] Smith, T., **Complete Family Health Encyclopedia**, Dorling Kindersley Limited, London, 1990.

quantities which are connected with one another by different kinds of operators. Coding the data base (preparation of the model vectors) has become a little bit more complicated as in other cases. Let us try to analyze the simple part of the decision tree, which is shown in Figure 6.1.

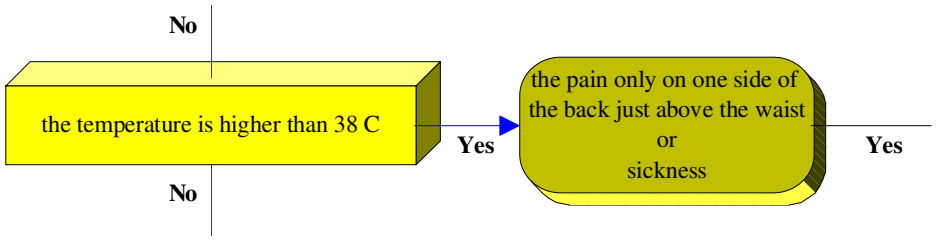

Figure 6.1: Part of the decision tree in case of diagnosis of the back pain.

How to solve the problem? We will take up the simplest possibility and use instead of one model vector two model vectors to describe the problem. We should also agree upon, how to label different quantities; the quantity has value "1", if it is real, and "0" otherwise.

According to the above convention, the case from Figure 6.1 can be written in form of three model vectors as follows:

{..., 1, 1,0, ...},

{..., 1, 0,1, ...}, and

{..., 0, ...}.

The first two records (model vectors) convey:

..., *IF*

the patient has the temperature higher than 38^{o}C and
he/she feels the pain just on one side of the back

or

he/she feels feebleness,

THEN ...

The third record (model vector) conveys:

..., *IF*

the patient has the temperature less than 38^{o}C,

THEN ...

A simple diagnosis in case of back pain is shown in Figure 6.2. The decision tree from this figure gives basic information for the preparation of knowledge base. Input parameters (symptoms of the disease) of model vectors are as follows:

1.1 the pain after lifting up a heavy burden, after a cough
and/or

1.2	after an exhausting physical exercise
2	the temperature is higher than 38°C
3.1	the patient is over 60 years old and/or
3.2	the patient spent several weeks in bed or in a wheel chair
4	the patient is over 45 years old
5	the pain is worse in the morning when getting up
6.1	the patient is hindered by the pain when moving or
6.2	the stitch in one leg
7	the pain is localized mainly in the back and does not spread anywhere else
8.1	the pain only on one side of the back just above the waist and
8.2	sickness
9	the pain is much worse on one side of the backbone
10	the pain occurs mostly in the neck or higher in the back between shoulder blades

Output parameters of the model vectors - diagnosis have the following meaning:

1	SCIATICA, caused by pressure on a root of a kidney nerve; consultation with the doctor
2	a possible LUMBAGO, resulting from various exertions
3	a possible infection of kidneys - PYELONEPHRITIS, or the pain might be accompanying general virus inflammation; consultation with the doctor
4	pains in the back (also severe ones) might be resulting from virus infections, e.g. INFLUENZA; consultation with the doctor
5	an immediate consultation with the doctor, a possible bone injury
6	a possible ARTHRITIS in the neck vertebrae; a consultation with the doctor
7	a possible OSTEOARTHRITIS in a lower breast, renal or back part of the backbone; a consultation with the doctor
8	the patient's bed might not give him enough support, a chronic inflammation of joints is possible, as a result of a SPONDYLITIS; a consultation with the doctor
9	a consultation with the doctor

NOTE: The data base also named the knowledge base in the given example, is written on the EXAMPLE6.CSV file. The entire phenomenon - diagnosis of the newly occurred back pain is presented with 18 model vectors. Each model vector has 23 parameters, where 14 of them are input parameters, and 9 of them are output parameters. Results shown in the next subsection are obtained on the basis of (prediction) vectors, written on the EXAMPLE6.PRD file.

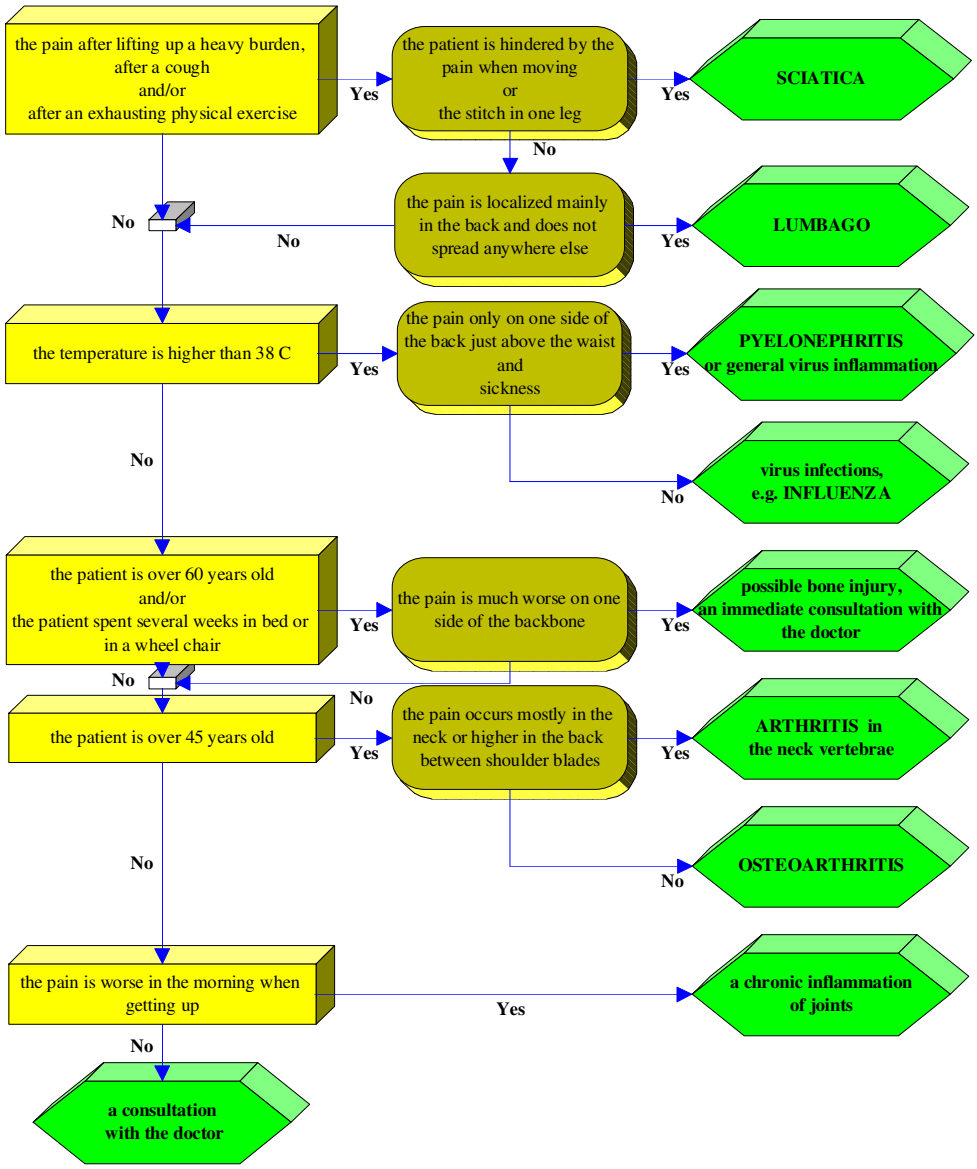

Figure 6.2: Decision tree in case of back pain diagnosis.

6.3 Verification of the Model

The model is not verified in the right sense of the word. Moreover, it is controlled just for a single value (0.5) of the penalty coefficient. For different combinations of symptoms of possible diseases are used as test examples. The first two are the first two samples from the knowledge base, while the other two are hypothetical samples as follows:

- pain appears after a patient has lifted up heavy furniture, after a cough,
  and
  after gymnastics
- a patient has temperature higher than 38°C,
- patient is over 60 years old

- patient has temperature higher than 38°C,
- patient is over 60 years old
  and
  he has spent a few weeks in bed recently

The results can be interpreted as certainties for each diagnosis, similar as we have done this in the case of diagnosis PCP during driving (see Example 5!). The test results show certainties are practically equal to one for SCIATICA in the first two samples (which are the right diagnoses); all other diagnoses have values practically equal to zero. As we can see, the model is able to predict the learning cases. (Users can check, if the model also predicts other samples from the knowledge base.)

The third hypothetical sample has about 50% certainty for SCIATICA, about 25% for LUMBAGO and about 25% for virus infection (e.g. INFLUENZA). The fourth hypothetical sample has about 50% certainty for virus infection (e.g. INFLUENZA) and about 50% for OSTEOPOROSIS. All other diagnoses would have values in case of greater penalty coefficient predicted in such a way, that the sum of certainties in each sample is exactly 1, or 100%!

6.4 Comments

Only the doctors can assess the validation of the aiNet model in the given example, so we leave the final assessment to the experts. It should be mentioned again that the given example is of illustrative nature. Practical use of similar and/or of more sophisticated models demands a user friendly interface for pre-processing the knowledge base and post-processing the predicted results. The data from decision trees will be automatically transformed in and out of binary form, which the aiNet uses for prediction.

The authors strongly believe that the aiNet is one of the powerful tools which can help to prepare highly sophisticated models in medical diagnosis. Such models can help doctors in the practice, especially in cases of more complex diseases. The technique is appropriate for coding and presentation of the knowledge, which has not been explicitly recorded yet. There are also some other possible applications, for example recognition of genetic codes, etc.

References

10. Carpenter, W., C. and Barthelemy, J., F., **Common misconceptions about neural networks as approximators**, Proceeding of the 3rd International Conference on the Application of Artificial Intelligence to Civil and Structural Engineering, in Neural Networks and Combinatorial Optimization in Civil and Structural Engineering (Topping, B. H. V., editor), pp. 11-18, 1993.

11. Hecht-Nielsen, R., **Neurocomputing**, Addison-Wesley Publishing Company, Inc., 1990.

12. Guyon, L. et al, **Comparing different neural network architectures for classifying handwriting digits**, Proc. of the Int. Joint Conf. on Neural Networks, **II**, pp. 127-132, IEEE Press, New York, June 1989.

13. Hecht-Nielsen, R., **Nearest matched filter classification of spatiotemporal patterns**, Applied Optics, **26** (10), pp. 1892-1899, 1987.

14. Yeh, Y., Kuo, Y. & Hsu, D**., Building KBES for diagnosis PC pile with artificial neural network**, Journal of Computing in Civil Engineering, Vol. 7, No. 1, pp. 71 - 93, 1993.

15. Smith, T., **Complete Family Health Encyclopedia**, Dorling Kindersley Limited, London, 1990.