This is Chapter 5 of the term paper:
Ulrike Nauck:
NEFCLASS-PC: Ein Neuro-Fuzzy-Klassifikationstool unter MS-DOS
Technical University of Braunschweig, 1997
(all other chapters are in German language)

Chapter 5

Manual for NEFCLASS-PC

NEFCLASS-PC is an interactive simulation software to develop, train, and test a Neuro-Fuzzy System for Classification. NEFCLASS-PC 2.04 is the fifth released version of the neuro-fuzzy classification software for MSDOS PC using an 80286 processor or better.

5.1 The Features of NEFCLASS-PC

NEFCLASS is a neuro-fuzzy model based on a generic 3-layer fuzzy perceptron, and it is used for classifying data (patterns). NEFCLASS is trained with a set of patterns, where each pattern belongs to one of a number of distinct classes (crisp classification). NEFCLASS finds fuzzy rules by scanning the data, and later optimizes these rules by learning the parameters of the fuzzy sets that are used to partition the domain of the input variables (features of the patterns).

After the learning process the resulting NEFCLASS system can be used for classifying new, previously unknown data. The system can be interpreted in form of fuzzy rules like

```
IF          x1 is A1
      and   x2 is A2
      and   x3 is A3
      and   x4 is A4
THEN  the pattern (x1,x2,x3,x4) belongs to class i,
```

where A1 - A4 are linguistic terms (e.g. small, medium, large) represented by fuzzy sets. This feature allows you to learn something about your data, and use this knowledge to get a smaller perhaps better solution (i.e. classifier).

Since NEFCLASS-PC is a tool that allows you to create a classifier based on simple linguistic rules you can use your own knowledge to initialize the system by feeding already known rules like the depicted one above into the network.

Interpretation of the results and use of prior knowledge are the main features of NEFCLASS-PC. If you are mainly interested in an optimal classification result, then you should keep in mind that there are very powerful statistical methods and neural networks.

Installation Guide

NEFCLASS-PC can be obtained

- from anonymous ftp at ftp.fuzzy.cs.uni-magdeburg.de in the directory /pub/nefclass, in the file nefclass.zip (compressed with PKZIP 2.04g) or
- via the World Wide Web at http://fuzzy.cs.uni-magdeburg.de/~nauck or
- E-Mail, Fax or letter to
  Dr. Detlef Nauck
  Otto-von-Guericke University of Magdeburg
  Institute of Information and Communication Systems
  Universitaetsplatz 2
  D-39106 Magdeburg
  Phone: +49.391.67.12700
  Fax: +49.391.67.12018
  E-Mail: nauck@iik.cs.uni-madeburg.de

For installation just copy the zip file to a new directory, and use "pkunzip -d nefclass" to uncompress it - thats all. The files included in the zip archive will be written to the current directory, and you will get a new directory "hpglplot" below it that contains another zip file (plt100c.zip). This additional archive holds the original distribution of the HPGL BGI driver from Ullrich von Bassewitz. His BGI driver is used by NEFCLASS-PC, and the terms of usage are, to distribute the original archive of this BGI driver, too. You do not need this zip file to run NEFCLASS-PC.

To use the graphical features of NEFCLASS-PC you need Borland's graphical interface drivers (BGI drivers). Included are the necessary drivers egavga.bgi, trip.chr, and litt.chr. The file plotter.bgi is needed to create HPGL files from the graphics of NEFCLASS-PC. If you already use BGI drivers for other programs, you will probably have the first three files. You can store the BGI drivers in a separate directory if you want to. But then you must set the DOS environment variable BGIPATH to the directory, where the drivers reside, so NEFCLASS-PC can find them there. If you don't use this environment variable, NEFCLASS-PC assumes the BGI drivers to be in the current directory.

To use NEFCLASS-PC, just start the executable nefclass.exe, its a DOS application. The user interface is using the TurboVision libraries of Borland, so it should be easy to use. It can also be used under Windows running in a DOS-Window.

The Iris Data Set

NEFCLASS-PC learns from training data, which must be provided in a pattern file. This manual uses the Iris data which is also distributed in combination with NEFCLASS-PC as a concrete example for the explanations. Iris data is perhaps the best known database to be found in the pattern recognition literature. Fisher's paper [FISHER36] is a classic in the field and is referenced frequently to this day [DUDA/HART73]. The three types of Iris flowers can be classified by the length and width of their sepals and petals. So the attribute information is:

- sepal length in cm
- sepal width in cm
- petal length in cm
- petal width in cm
- class: Iris Setosa
  Iris Versicolour
  Iris Virginica

The data set contains 3 classes of 50 instances each, where each class refers to a type of Iris plant. These are 150 cases of 4 numeric predictive attributes and the class coded as a binary vector of 3 components. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other. Here are some Statistics on the data set:

	Min	Max	Mean	SD	Class Correlation
sepal length	43	79	584	83	0.7826
sepal width	20	44	305	43	-0.4194
petal length	10	69	376	176	0.9490 (high!)
petal width	1	25	120	76	0.9565 (high!)
No missing attribute values					

The Iris Data Files

IRIS.DAT Pattern file containing the complete Iris data set sorted and reformatted for use in NEFCLASS-PC 2.0. This file contains the whole data set with 150 cases, 50 for each class. The outputs are coded as follows:

- class 1 0 0 is Iris-setosa,
- class 0 1 0 is Iris-versicolor,
- class 0 0 1 is Iris-virginica.

This is a part of the original head and two lines of data out of IRIS.DAT:

```
% This are the 150 patterns,
% first given are the number of patterns, inputs and outputs
PATTERNS
150 4 3
5.1  3.5  1.4  0.2  1 0 0
4.9  3.0  1.4  0.2  1 0 0
```

IRIS1.DAT	Pattern file containing one interleaved half of the Iris data.
IRIS2.DAT	Pattern file containing the the other interleaved half of the Iris data.
IRISNOCL.DAT	Pattern file containing the Iris data without class information. Again a part of the original head and data lines:

```
% This are the 150 patterns,
% first given are the number of patterns, inputs and outputs
% (here: 0)
PATTERNS
150 4 0
5.1  3.5  1.4  0.2
4.9  3.0  1.4  0.2
```

To classify unknown patterns, they must be stored like the data in this file (see also p. 71).

IRIS.ORG	Original Iris data without any head or coding (not for use with NEFCLASS-PC). Again the first two data lines.

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
```

5.2 How to use NEFCLASS-PC

There are two possibilities to create a manual for a software tool. One is to describe the menue in the sequence of listing with the problem of perhaps describing some functions without the background of others. The other one is a “guided tour” through the program with the problem of hopping through the menue. For the benfit of understanding how NEFCLASS-PC works the decision went for the last one. The structure of the menue with page numbers of further description is depicted in the following table.

Menue

File	**N**etwork	**P**attern	**R**un	**E**dit	**W**indow	**H**elp
New	**N**ew F4 *p. 44*	**L**oad... F7 *p. 47, 58, 60*	Learning **C**ontrol *p. 48*	**U**ndo	**T**ile	**H**elp Index *p. 77*
Open... F3 *p. 56, 60*	**C**hange *p. 65*	**D**isplay *p. 47*	Learning **P**arameters *p. 49*	Cu**t** Shift + Del	C**a**scade	**A**bout
Save F2	**D**isplay *p. 46*		Start **L**earning F8 *p. 53*	**C**opy Ctrl + Ins	Cl**o**se all	**L**icense
S**a**ve as	**L**oad... F6 *p. 64*		**T**est Network *p. 56, 58*	**P**aste Shift + Ins	Size/Move Ctrl + F5	**M**emory
Save a**l**l	**S**ave F5 *p. 56*		**S**tatistics *p. 60*	C**l**ear Ctrl + Del	Zoom F5	
Change dir...	S**a**ve as... *p. 56*			**S**how Clipboard	Next F6	
DOS shell	**E**dit Rules *p. 58, 67*				Previous Shift + F6	
E**x**it Alt+X	Save **R**ules *p. 59*				Close Alt + F3	
	Des**c**ription *p. 56*					

The Program Handling

The commands in NEFCLASS-PC are issued by using the mouse. Move the mouse pointer on the menu, or button described and click the left button to execute the command. The commands can also be executed by pressing keys.

In this description all commands or buttons are written in capital letters. A command that is invoked via the menu will be written in a short form like FILE|EXIT. This means you must select the menu entry FILE, open the entry, and select the command EXIT.

If you want to use keys, consider the following descriptions:

- menu bar inactive
  - press F10 or for activation or
  - press ALT plus the highlighted letter of a menu entry for activation and opening pull down menue
- menu bar active
  - cursor (right, left) to select a menue and press Enter to open a pull down menue or
  - press the highlighted letter of a menu entry to open a pull down menue
- pull down menue opend
  - curser (down, up) to select function and press Enter to execute a menu command or
  - press the highlighted key to execute a menu command
- buttons of dialog boxes (green)
  - select by pressing TAB (forward) or Shift+TAB (backward) and press Enter or Space to execute
- fields of dialog boxes
  - edit fields (blue)
  - list fields (petrol)

  To edit a field not connected to a list just select it by pressing TAB (forward) or Shift+TAB (backward) till the cursor is in edit position, edit the field and select next one, or select by pressing the highlighted key (not possible if cursor in edit position). If you change a list, select the list by pressing TAB (forward) or Shift+TAB (backward), choose the list field by cursor up/down, then move to the input area. and confirm with the “set” button. Try this in the NETWORK|NEW window during the guided tour (go by TAB till “Class names” is highlited in white letters. Then choose the line to be changed by cursor up/down. You will see that the entries on the right change. Afterwards press Shift+TAB two times to reach this edit field. Edit and leave by TAB to the “set” button. Don’t use Enter for confirmation of field input. This means confirmation of the window input and closes it at the same time). However, it’s much easier to use a mouse.

- decision fields of dialog boxes
  select by pressing TAB (forward) or Shift+TAB (backward), decide by cursor which line or column should be active.
- change DOS mode to Windows mode (if NEFCLASS-PC runs under Windows)
  press ALT + Enter
- Escape
  ESC closes last opened dialog box without change
- exit program
  - press ALT+X or
  - select the menu entry FILE and then EXIT (FILE|EXIT).

  any dialog box must be closed before (usually by pressing on OK, CLOSE, or CANCEL). Editor windows will be closed automatically.
- Help
  At any time you can press F1 or a help button to obtain context sensitive help information.
- status bar
  List of function keys of frequently used commands to use as hot keys
  F1 Help **F4** NewNet **F5** SaveNet **F6** LoadNet **F7** LoadPat. **F8** Learn **Alt+X** Quit

As mentioned NEFCLASS-PC is a program for DOS but can also be started under Windows. All screendumps shown here are created under Windows and have the typical Windows frame with a button for changing to fullscreen mode under DOS. While showing graphical displays NEFCLASS-PC also changes to fullscreen mode. You can change back to Windows mode by ALT + Enter.

5.3 Guided Tour through NEFCLASS-PC - *Create, Train and Test*

This chapter is the first one of four chapters building a “guided tour”. To explain what should be learned from these four parts let me reflect the problem NEFCLASS-PC should be able to solve:

Your Problem
- There is data to be classified.
- For some reason (e.g. you need a fast, cheap solution, you didn’t find another one, or you are just interested) you want to do a classification with a Neuro-Fuzzy System.
- You want to know how successfull the classification was, so you want to know the performance of the Neuro-Fuzzy System.
- You want to modify the Neuro-Fuzzy System in order to find out if the performance can be improved.

- In the case you have prior knowledge about the data, you want the Neuro-Fuzzy System to use it
- You want to be able to interpret the results

You need

- Data with classification information devided into two parts
  - one part for training the network (here IRIS1.DAT)
  - one part for testing the performance of the trained network (here IRIS2.DAT)
- A software tool like NEFCLASS-PC
  - to create, train, and test a network
  - to give you graphical and textual displays for interpreting the results
  - to do some statistics to get prior knowledge about the data..

So in the first part of the guided tour

- a new network (p. 43) is created,
- a pattern set (p. 47, 58, 60) is loaded, here IRIS1.DAT
- the network (p. 48) is trained and afterwards saved (p. 56).
- the performance of this trained network is first to be tested with the learning data (p. 56) and then with the second part of data here IRIS2.DAT (p. 58).
- the rulebase is to be inspected (p. 58)
- the statistic procedure is used in order to find informations that give hope to improve the performance when changing the network parameters (p. 60).

In the second part of the guided tour

- the number of rules is changed in the trained network saved in part one (p. 65)
- the number of fuzzy sets is changed in the trained network saved in part one (p. 66) because of the hints given by the statistics
- in every case the network has to be relearned and tested with IRIS1.DAT and IRIS2.DAT, respectively.

In the third part of the guided tour

- new networks are created by using prior knowledge (p. 67)
- the networks are analogously trained and tested to the network in part one

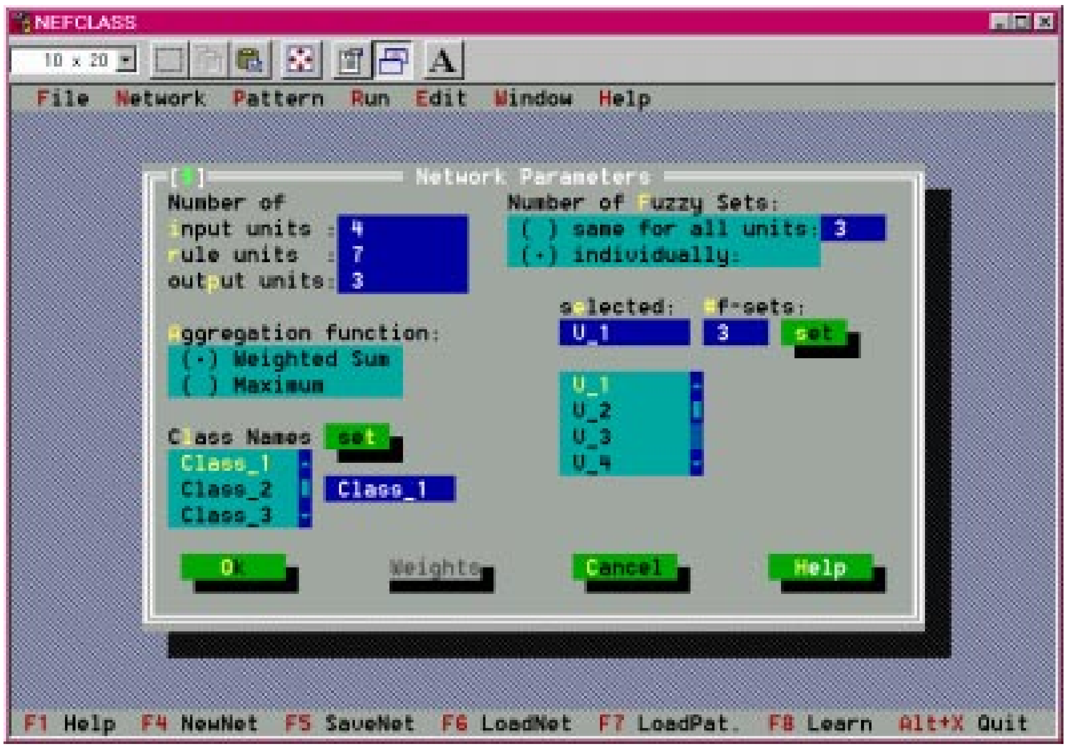
In the fourth part of the guided tour

- it is described how to write NEFCLASS pattern files (p. 71)
- it is described how to understand and edit the NEFCLASS network file (p. 74).

5.3.1 Create a New Network

After starting NEFCLASS-PC (by invoking nefclass.exe) you will see an information box which can be closed by pressing Enter or Space or klicking the OK button. This causes the opening of the licence window which should be read and then also be closed. The resulting screen only shows the menue bar at the top and the status bar at the bottom.

For the definition of a new network use the NETWORK menu, and the command NEW (NETWORK|NEW), or by just pressing F4. If there is currently a network defined, a dialog box will appear, and ask you if it is ok to delete the current network, and the current set of patterns, if a pattern file was loaded. Then it is your decision, if you really want to start with a new network or to continiue with the previously defined one. For the guided tour you should use a new network and the command will cause the following screen.


NEFCLASS
10 x 20
File Network Pattern Run Edit Window Help
Ok
Weights
Cancel
Help
F1 Help F4 NewNet F5 SaveNet F6 LoadNet F7 LoadPat. F8 Learn Alt+X Quit

```
Network Parameters
Number of                          Number of Fuzzy Sets:
input units  : 4                   ( ) same for all units: 3
rule units   : 7                   (•) individually:
output units : 3
                                      selected:   #f-sets:
Aggregation function:                 U_1         3          set
 (•) Weighted Sum
 ( ) Maximum                          U_1
                                      U_2
Class Names  set                      U_3
 Class_1                              U_4
 Class_2       Class_1
 Class_3
```

With help of this dialog window, where default values are given, the new network can be defined:

- **Number of**
  - input units (= variables):

    This number (max. 50) must be equal to the number of features of your patterns (component of a vector) which you want to classify (4 for Iris data).

- ○ rule units:

  Here you specify the maximal number of rules your network is allowed to have. The rules are created during the learning algorithm, or can be entered by yourself. The maximum number is 500, but please note, that the program becomes terribly slow if you allow to create such a lot of rules. Usually You will be successful with a substantially smaller number. This depends, of course, on the application, and has to be tested out. On the other hand the learning algorithm might not create this number of rules you specified, it is possible, that the trained network has less rule units.

- ○ output units (= classes):

  This number (max. 50) must be equal to the number of classes of your patterns which you want to use for training (3 for Iris data).

- **Aggregation function**

  This is the activation function for the units of the output layer.

  - ○ "Weighted Sum" means that the mean of the connected rule unit activations is used as activation for the output units. If you decide to use rule weights this will be a weighted mean.
  - ○ "Maximum" means that only the maximum of the activations of the connected rule units will be used (eventually multiplied by an optional rule weight).

- **Class Names**

  Here you can specify individual names for the output units (classes). By default an output unit j has the name Class_j. In the list there are always names for 50 units (the maximum number of units), but only the first k will be used, where k is the number of output units specified by you. A name may have up to 8 characters.

- **Number of Fuzzy Sets**

  For each input unit a number of fuzzy sets (a fuzzy partitioning) has to be defined. There are triangular membership functions used that are equally distributed on the domain of the respective input feature. The leftmost and rightmost functions are shouldered. The fuzzy sets automatically get names.You have the following options:

  - ○ “same for all units”

    means that you can select the same number of fuzzy sets for each variable by entering the desired number into the input field next to it. If you select this option the individual numbers that you may have specified in the list (see below) are ignored. But the names you may have changed in the list are always used (see below).

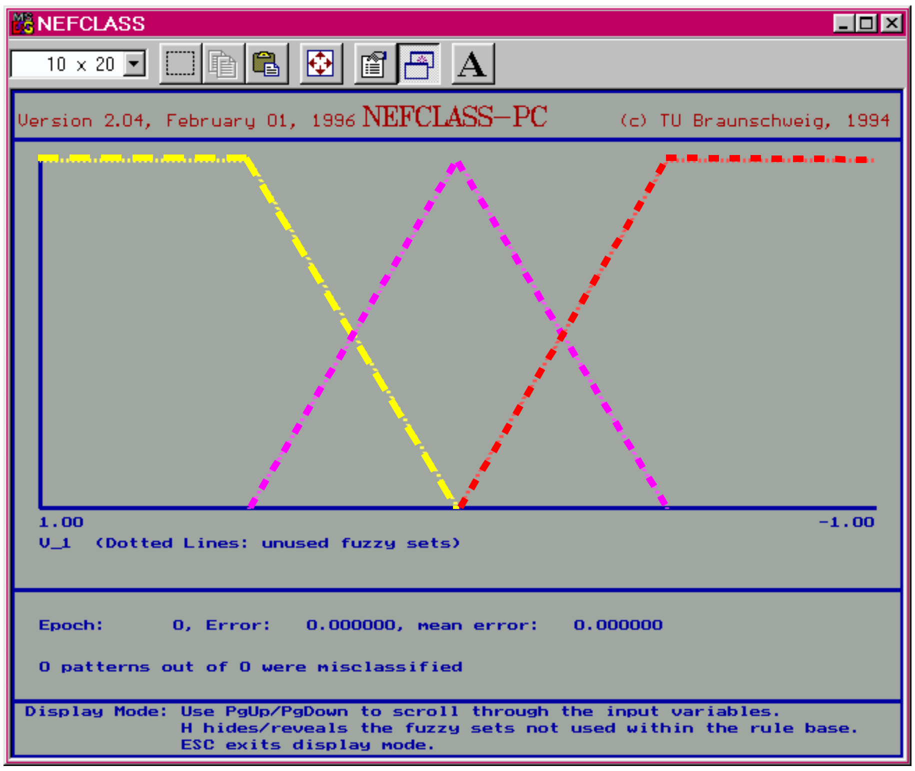
  - ○ “individually”

    means that you can define an individual number of fuzzy sets for each input unit. If you choose the latter option, you can select a unit from the list of input unit names, and define a new number of fuzzy sets for the selected unit, and - if you want to - you can also specify a new name for this unit. A name may have up to 8 characters. Each variable can have a maximum of 7 fuzzy sets, and it must have at least 1. If you specify only one fuzzy set for a variable, this means that it is ignored for the classification of patterns, because if yields a membership value of 1 for all input values. This list also carries 50 entries, from which only the first k are used, k being the number of input units specified by you.

For the Iris data example there is no need to change anything, just click on OK. You will get a NEFCLASS network with 4 input units, and 3 output units, the network may learn a maximum of 7 rule units, and for each input variable 3 fuzzy sets are automatically defined by equally distributed triangular membership functions.

5.3.2 Graphical Representation of the New Network

For displaying the membership functions, select NETWORK|DISPLAY, and the same dialog box opened for the definition of the network will open again. This time there cannot be changed anything, the entries can only be viewed. But the button WEIGHTS has green colour now and can be chosen. Clicking on WEIGHTS, will open a graphical representation of the fuzzy sets.



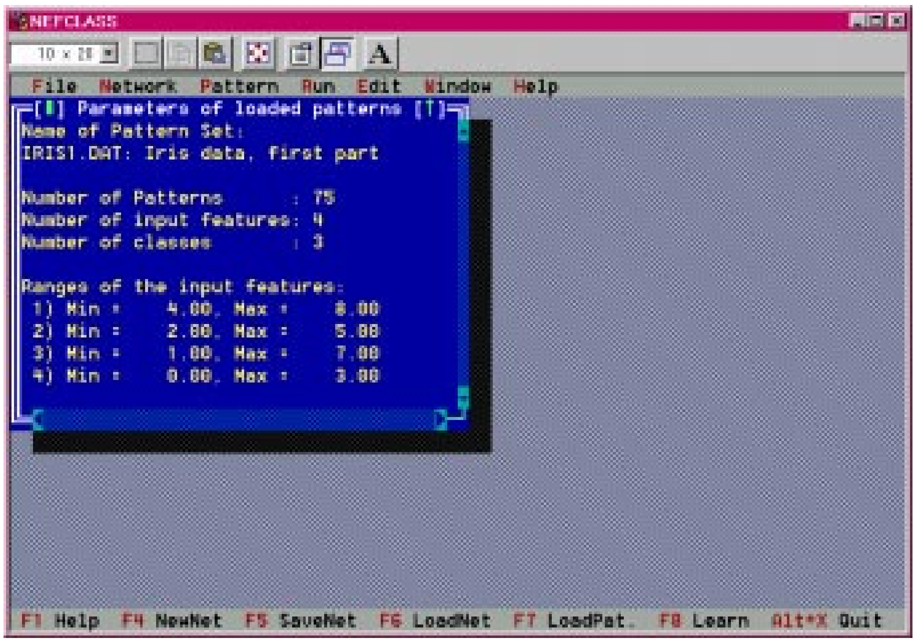
```
Version 2.04, February 01, 1996                (c) TU Braunschweig, 1994

V_1  (Dotted Lines: unused fuzzy sets)

Epoch:        0, Error:    0.000000, mean error:    0.000000

0 patterns out of 0 were misclassified

Display Mode: Use PgUp/PgDown to scroll through the input variables.
              H hides/reveals the fuzzy sets not used within the rule base.
              ESC exits display mode.
```

The fuzzy partitions of the different input units can be display by using the PgUp and PgDown keys. All fuzzy sets are drawn with dotted lines. This means that they are not used in any rule represented in the network. This is due to the fact that the network currently doesn't have any rules. They will be created during the learning procedure. Furthermore the domain for each variable is the real interval [-1,1]. These are default values which are changed, after a pattern set was loaded, and the training procedure has started. To leave the graphics press ESC.

5.3.3 Train the Network

There must be patterns to train a network. A pattern file will be loaded by using PATTERN|LOAD (or F7). If there are already patterns loaded, a dialog box will appear, and ask you whether it is ok to delete the current patterns. By default only files with the extension ".dat" are displayed in the list of files. For the Iris data example search for the file IRIS1.DAT in the file dialog box, and press OK. Also press OK in the following message box that tells you that the patterns have been loaded successfully.

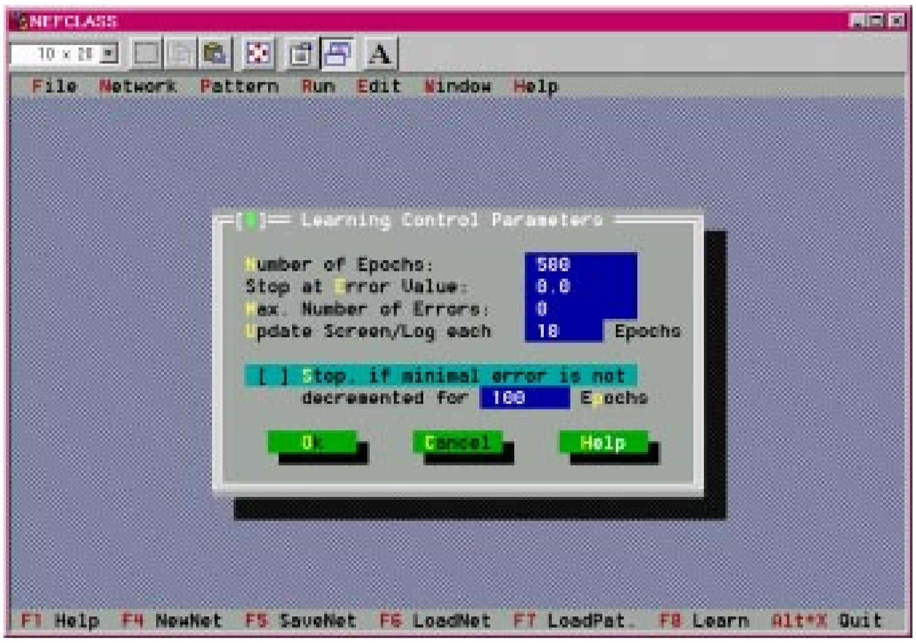
You may then view the pattern parameters with the PATTERN|DISPLAY command in a text window.


NEFCLASS
10 x 28
File Network Pattern Run Edit Window Help
F1 Help F4 NewNet F5 SaveNet F6 LoadNet F7 LoadPat. F8 Learn Alt+X Quit

```
[■] Parameters of loaded patterns [↑]
Name of Pattern Set:
IRIS1.DAT: Iris data, first part

Number of Patterns       : 75
Number of input features: 4
Number of classes        : 3

Ranges of the input features:
 1) Min =    4.00, Max =    8.00
 2) Min =    2.00, Max =    5.00
 3) Min =    1.00, Max =    7.00
 4) Min =    0.00, Max =    3.00
```

To close the text window, just click on the close box in the upper left corner, or press Alt+F3.

To train the network with the loaded pattern file, there is need for a little more input. The learning process is guided by a few learning parameters. First take a look at the learning control options, by invoking the command RUN|LEARNING CONTROL.


NEFCLASS
File Network Pattern Run Edit Window Help
Ok
Cancel
Help
F1 Help F4 NewNet F5 SaveNet F6 LoadNet F7 LoadPat. F8 Learn Alt+X Quit

```
Learning Control Parameters

Number of Epochs:              500
Stop at Error Value:           0.0
Max. Number of Errors:         0
Update Screen/Log each         10     Epochs

[ ] Stop, if minimal error is not
    decremented for   100     Epochs
```

In the dialog box some options can be defined, which guarantee that the algorithm will stop after a reasonable time and/or at a good classification result.

- **The number of epochs** (cycles or sweeps through the pattern set)

  After how many epochs learning has to be stopped, i.e. for how many times the pattern set shall be presented to the network for learning.

You can also define three different stop criterions depending on the error

- **Stop at Error Value**

  Stop learning, when a specific error value is reached. Due to the mathematics involved you cannot reach an error value of 0.0 (eventually you can, if you use rule weights). The error is also depending on the number of rules. You should use this option only, if you have gained some experience with the program.

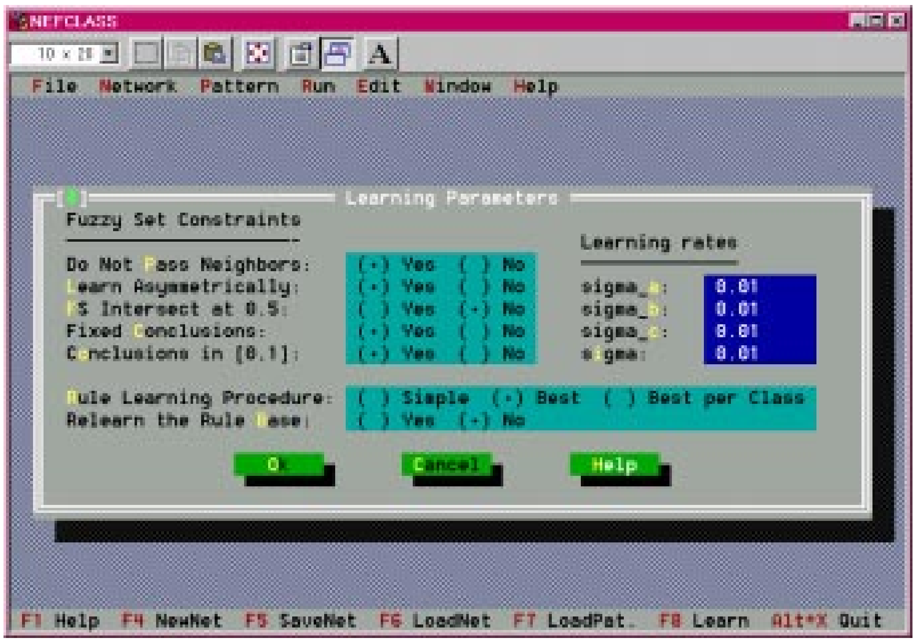
- **Max. Number of Errors**

  Define the maximum number of admissable misclassifications. If you have - let's say - 100 patterns, and you want to have a 95% correct classification, then you could enter 5 for this option. Unfortunately, the learning procedure is heuristic, and it cannot be guaranteed that the error (or the number of misclassifications) becomes less than any given value.

- **Stop if minimal error is not decremented for a given number of epochs**

  This is usually the best stop criterion: For this example please select this option with the mouse (or with the TAB key, and then pressing the space bar), an X appears between the brackets and marks it selected. Then enter 50 in the text field (select the text field with the mouse, or with the TAB key). It will be checked whether the learning process has to be stopped, because the error did not become smaller during the last 50 epochs.
- **Update screen/log**, the update rate for the screen and the protocol file during learning. The default value 10 is ok for this demo.

The dialog box can be closed now by clicking on the OK button. The next thing is to take a look at the learning parameters by using the command RUN|LEARNING PARAMETERS.


NEFCLASS
10 x 28
File Network Pattern Run Edit Window Help
OK
Cancel
Help
F1 Help F4 NewNet F5 SaveNet F6 LoadNet F7 LoadPat. F8 Learn Alt+X Quit

```
Learning Parameters
Fuzzy Set Constraints                                        Learning rates

Do Not Pass Neighbors:        (•) Yes   ( ) No              sigma_a:   0.01
Learn Asymmetrically:         (•) Yes   ( ) No              sigma_b:   0.01
FS Intersect at 0.5:          ( ) Yes   (•) No              sigma_c:   0.01
Fixed Conclusions:            (•) Yes   ( ) No              sigma:     0.01
Conclusions in [0,1]:         (•) Yes   ( ) No

Rule Learning Procedure:      ( ) Simple   (•) Best   ( ) Best per Class
Relearn the Rule Base:        ( ) Yes   (•) No
```

In this box you will find the following entries

- **Do Not Pass Neighbors**
  - YES (default) means that during the learning process a fuzzy set must not pass its left or right neighbors. This is enforced by checking that **all three** parameters of a triangular membership function (left spread, center, right spread) stay smaller (larger) than the parameters of its right (left) neighbor.
  - NO means, that the fuzzy set may pass each other. This can lead to partitionings that cannot be interpreted, but sometimes give better classification results.

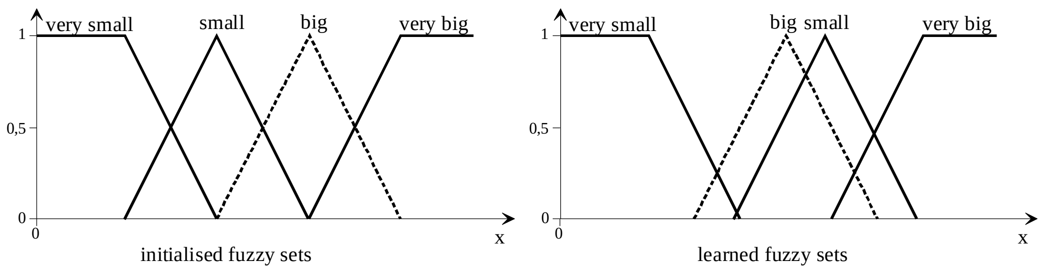


Figure 5.1: Fuzzy sets before (left) and after (right) learning process with window parameter set to “NO”. In fact the fuzzy set “big” is surrounded by all three parameters of the triangular membership function “small”. But even switching one point stands for “passing neighbours”

- **Learn Asymmetrically**
  - YES (default) means that during the learning process only at side of the triangular function is changed, where the input is located, i.e. either the left spread, or the right spread is changed.

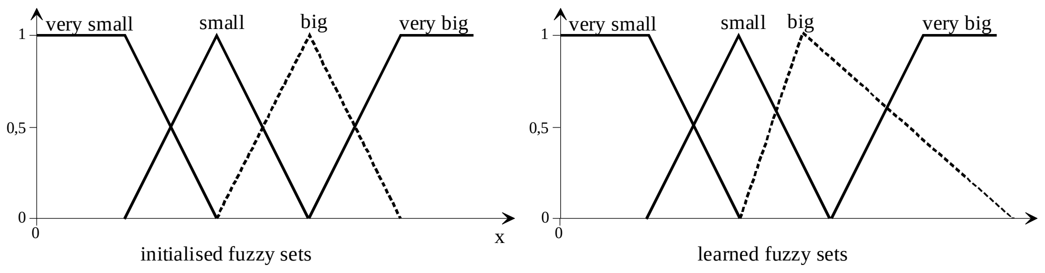


Figure 5.2: Fuzzy sets before (left) and after (right) learning process with window parameter set to “YES”.

  - NO means that both spreads are equally changed.

- **FS Intersect at 0.5**
  - YES means that two adjacent fuzzy sets must always intersect at a membership value of 0.5, i.e. for each value of the domain the sum of membership values over all fuzzy sets is always equal to one.
  - NO (default) means this restriction is not valid during learning.

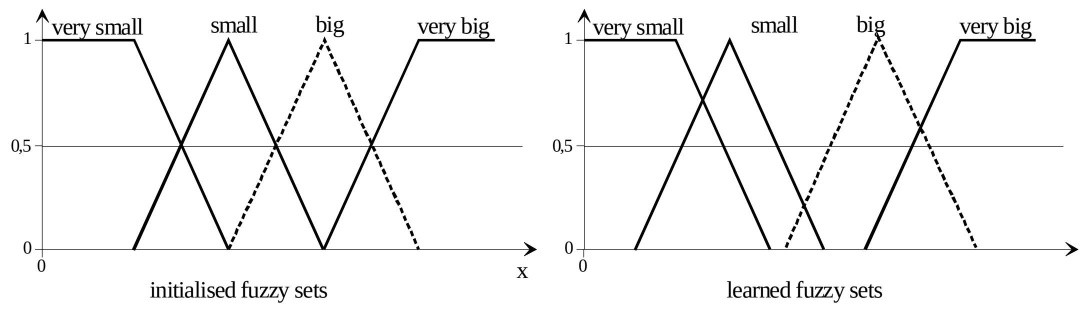


Figure 5.3: Fuzzy sets before (left) and after (right) learning process with window parameter set to “NO”. Here the fuzzy sets change symetrically in Figure 5.2 they change asymmetrically.

- **Fixed Conclusions**
  - YES (default) means NOT to learn the conclusion weights, they are fixed at 1.0.
  - NO means learning the conclusion weights.This is an option normally not used because this can lead to results that cannot be interpreted.

- **Conclusions in [0,1]**

  This option is only considered, if "Fixed Conclusions" is set to NO.
  - YES(default) means that during learning the conclusion weights will always be kept in [0,1].
  - NO means the conclusion weights may assume any value.

- **Learning Rates**
  - sigma_a, sigma_b, and sigma_c are the learning rates for the respective parameters of the triangular fuzzy sets.

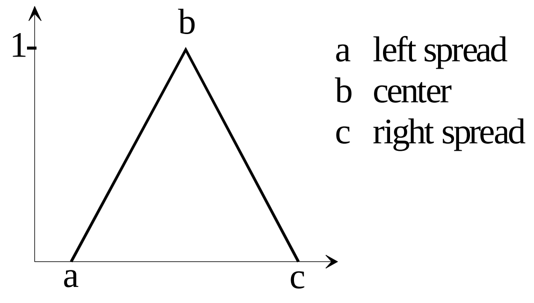


Figure 5.4: Triangular fuzzy set and its parameters

  - sigma is the learning rate for the conclusion weights This option is only considered, if "Fixed Conclusions" is set to NO.

- **Rule Learning Procedure**
  - “Simple” means that creation of rule will be stopped when the maximum number of rule units are learned, as specified with the NETWORK|NEW command.

    With this option the quality of rule learning depends a lot on outer circumstances.

    - Stopping when maximum number of rules is reached does not mean that the best rules has been created. The algorithm may stop before before all patterns are propagated and unusual patterns or patterns from different classes may not built rules then.
    - Therefore rule learning can only be successful, when the patterns are listed in random sequence in the pattern file for that the chance is growing that patterns of different classes can create rules.
    - This is also the reason why this procedure is very dependend on the sequence in which the patterns are presented in the file.

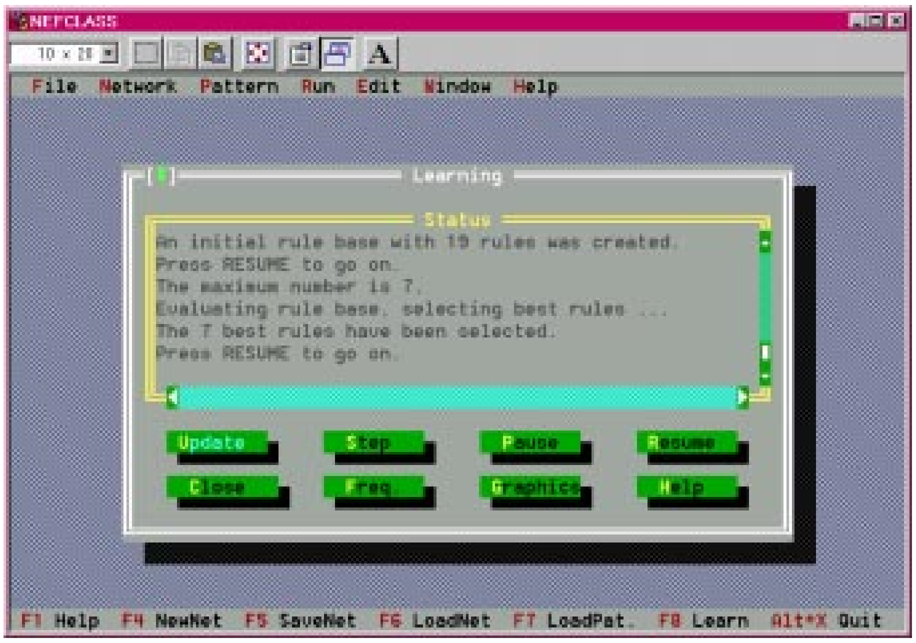
    If the rules are relearned because you changed the network, or checked the relearning option (see below), and if the network currently has more rules than allowed, then a sufficient number of rules is deleted from the end of the rule list.
  - “Best” means that all patterns are propagated three times for finding the best rules and that by this the rule learning procedure is made independent from the sequence of patterns.
    - At first as many rule units will be created as the pattern set demands, or as the program allows (500 in this version).
    - After this the rules will be evaluated. If it is found during the rule evaluation that a rule has a bad antecedent-conclusion combination, the conclusion of this rule is changed. By this rule learning is made independent from the sequence of patterns.
    - Then the rules will be evaluated a second time, and only the best rules will be kept, not exceeding the maximum number of rule units, as specified with the NETWORK|NEW command.
  - “Best per Class” means nearly the same as the option “Best”. Nothing changes in the first two steps. It is also independent from the sequence of patterns. But with the option “best” it is possible that the best rules are only for one class and that the remaining classes are not considered. In “Best per Class” the second evaluation step selects the best M rules for every class where M = N div L3, with N being the maximum number of rules, L3 being the number of output units, and div being the integer division (e.g. 8 div 3 = 2).

Choosing either this or the preceeding option is strongly recommended. The option “Best” is better, if you suspect that each class is best described by a different number of rules. The option “Best per Class” is better, if you suspect that an equal number of rules is needed for each class.

- Relearn the Rule Base
  - YES means that the rule learning procedure is invoked even if the network already has a rule base that was either created by the learning procedure, or defined in the network file from which the network was loaded.
  - If you select NO, then the rule base is only learned, if you define a new network (either without rules, or with prior rules entered by yourself), after you changed the maximum number of rules, or after you changed a fuzzy partitioning.

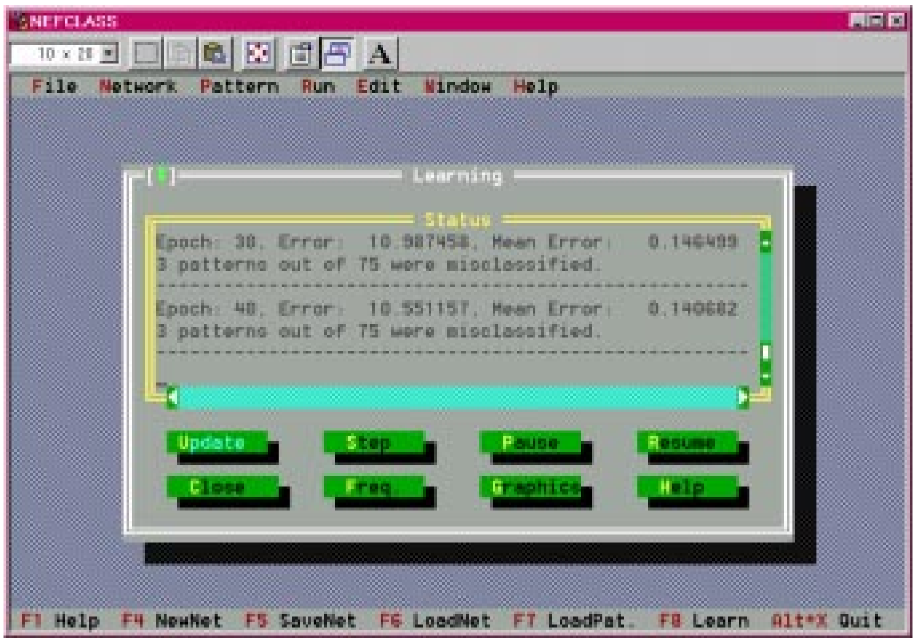
For continuing the “guided tour” there is nothing to be changed in this dialog box. Just click on OK. Now the learning process can be started: by using the command RUN|START LEARNING (F8). A dialog box will open, and you can watch the learning process.

In a first window the program tells you that 75 patterns have been trained and 19 rules have been created. With a click on the RESUME button of this window, learning goes on. After this a second message tells you that the 7 best rules have been selected. In this example the network must not have more than 7 rules, as previously defined by the NETWORK|NEW command.


NEFCLASS
10 x 18
File Network Pattern Run Edit Window Help
Learning
Status
Update
Step
Pause
Resume
Close
Freq.
Graphics
Help
F1 Help F4 NewNet F5 SaveNet F6 LoadNet F7 LoadPat. F8 Learn Alt+X Quit

```
An initial rule base with 19 rules was created.
Press RESUME to go on.
The maximum number is 7.
Evaluating rule base, selecting best rules ...
The 7 best rules have been selected.
Press RESUME to go on.
```

Of course, you can have more rules. Up to 500, if you want to, and if you have a real lot of time for training the network. Another click on the RESUME button will resume the interrupted learning process. You will now see the performance of the network as learning goes on.


NEFCLASS
File Network Pattern Run Edit Window Help
Learning
Status
Update Step Pause Resume
Close Freq. Graphics Help
F1 Help F4 NewNet F5 SaveNet F6 LoadNet F7 LoadPat. F8 Learn Alt+X Quit

```
Epoch: 30. Error:  10.987458, Mean Error:   0.146499
3 patterns out of 75 were misclassified.
--------------------------------------------------------
Epoch: 40. Error:  10.551157, Mean Error:   0.140682
3 patterns out of 75 were misclassified.
--------------------------------------------------------
```

This dialog box displays the status of the learning process. In the text window you can see the number of misclassified patterns, the error, and the mean error of the last documented epoch. The buttons of the window are explaind below:

- **Update**

  Displays information on the current epoch in the text window.

- **Step**

  Switch to/from single step mode. In single step mode you must press update, to issue another step (epoch) of the learning process.

- **Pause**

  Interrupt the learning process.

- **Resume**

  Go on with the learning process after an interruption. An interruption occurs, after the rules have been learned (twice if you choose another than the simple rule learning procedure), or when you have pressed the PAUSE button.

- **Close**

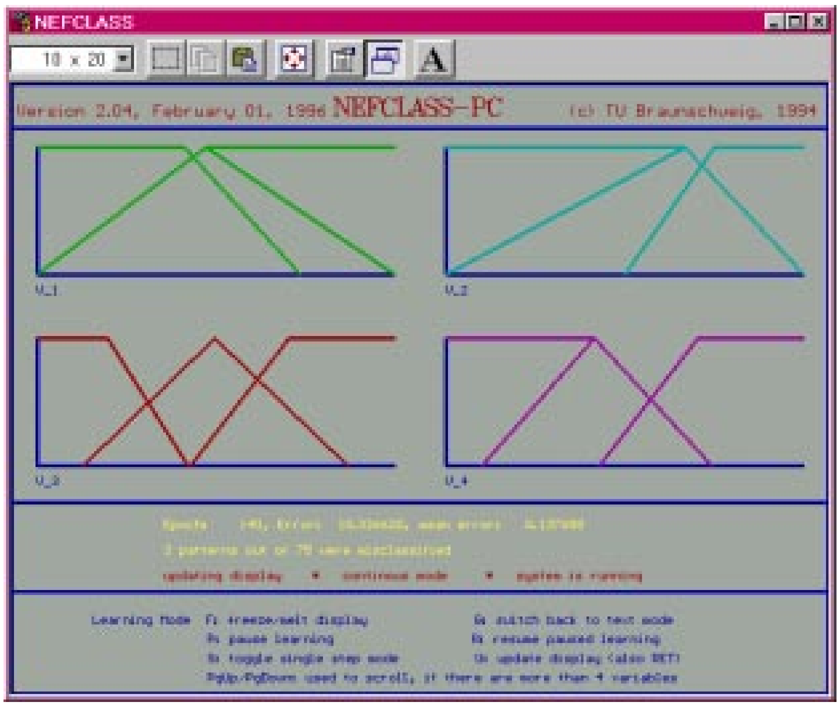
  Close the dialog window, when the learning period is over, or after you have interrupted the learning process by pressing PAUSE.

- **Freq**.

  Change the update frequency for the text or graphics display

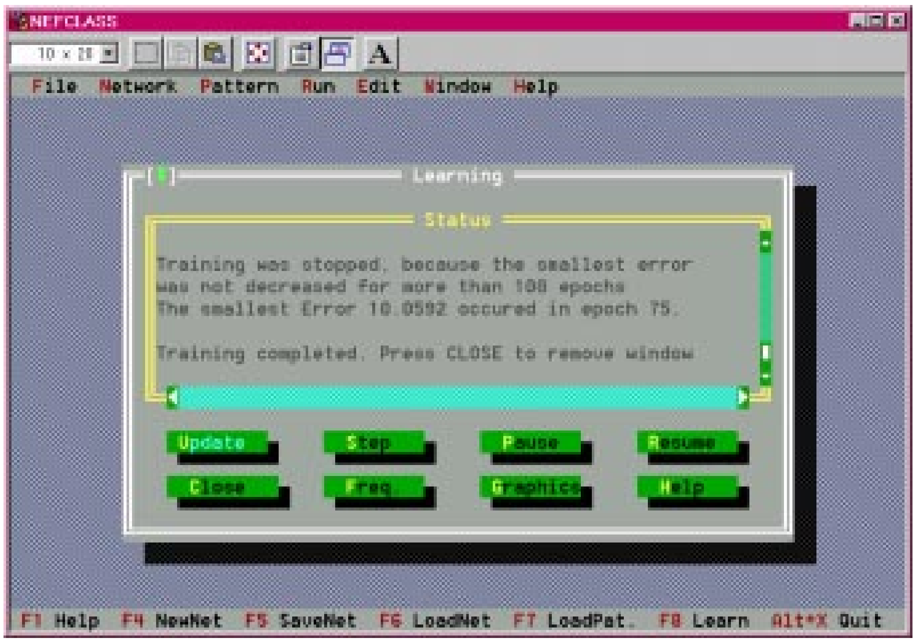
- **Graphics**

  Switch to the graphical display of the learning process. You will see the fuzzy sets of the input variables, and how they change.Press G to go back to text mode, or wait until learning stops, which will take you back to text mode automatically.



```
Version 2.04, February 01, 1996 NEFCLASS-PC        (c) TU Braunschweig, 1994

Epoch:   140, Error: [illegible], mean error: [illegible]
3 patterns out of 75 were misclassified
updating display   *   continuous mode   *   system is running

Learning Mode  F: freeze/melt display           G: switch back to text mode
               P: pause learning                R: resume paused learning
               S: toggle single step mode       U: update display (also RET)
               PgUp/PgDown used to scroll, if there are more than 4 variables
```

If you want to enjoy the the graphical display for this example you have to hurry, because learning will stop after only 126 epochs. The smallest error value will occur in epoch 75, and the network will misclassify 3 patterns.


NEFCLASS
10 x 20
A
File Network Pattern Run Edit Window Help
Learning
Status
Update
Step
Pause
Resume
Close
Freq
Graphics
Help
F1 Help F4 NewNet F5 SaveNet F6 LoadNet F7 LoadPat. F8 Learn Alt+X Quit

```
Training was stopped, because the smallest error
was not decreased for more than 100 epochs
The smallest Error 10.0592 occured in epoch 75.

Training completed. Press CLOSE to remove window
```

After learning has stopped press CLOSE to leave the learning dialog box.

The Program has created a trained network which can be saved now by using NETWORK|SAVE (F5) or NETWORK|SAVE AS. Because your network was not saved yet, both commands will behave equally. A file dialog box will open, where you must enter a filename (e.g. mynet.net), or select one from the list, where by default only files with the extension ".net" are displayed. Press OK in this dialog then, and again in the following message box. Apart from the guided way in this example please note that if the network was already saved, or if it was loaded from a file, the save command will write it to the same file as before, thus overwriting the existing file, no questions asked. The same happens if the network was read from a file, the filename is suggested again, and you can accept it by clicking on OK, but the existing files will be overwritten without notification.

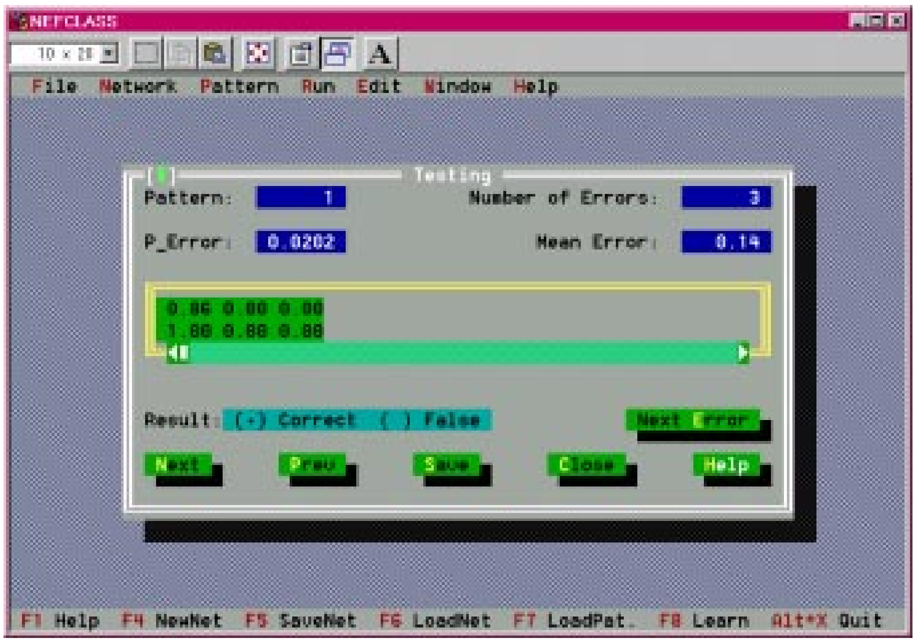
The saved file can be viewed by the FILE|OPEN command. A little ASCII editor will be invoked where any ASCII file can be handled. If you want to have a private description in this file you can insert it before saving the network by invoking the NETWORK| DESCRIPTION command. In this dialog window you can enter a one line description (max. 80 characters) to describe the current NEFCLASS network. The description will not be read from the network file, when you load the network again. So if you want to save a loaded network again, you have to specify a new description.

5.3.4 Test a trained Network and Lern how to Inspect the Rulebase

To check the performance of the trained network, it can be tested. First step is to take the data used for training, so you can see where the errors occur while training. Afterwards other pattern sets can be used.

Test With Training Data

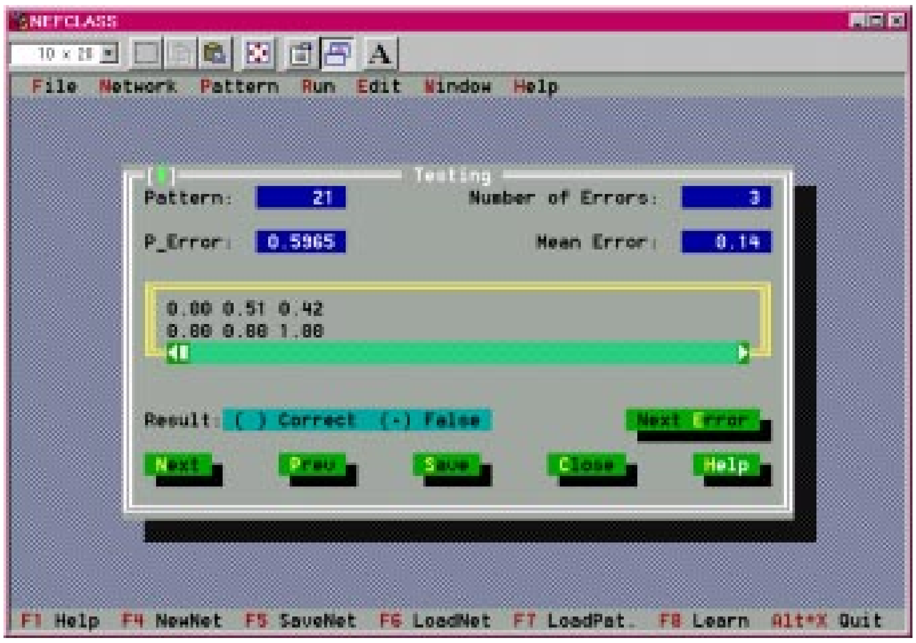
Use RUN|TEST NETWORK to invoke the test dialog box. In this Iris data example 3 errors have occured.The patterns that are misclassified can cyclically be inspected, by pressing on NEXT ERROR (the patterns 21, 60 and 62 are classified incorrectly). In the text area the network output, and below it the desired output specified by the patterns (i.e. the classification) is displayed. By using the buttons PREV or NEXT all patterns as well as the outputs can be inspected cyclically. The field “Result” shows whether the classification was correct (shown in the first of the next two screen copies), or false (shown in the second one).


NEFCLASS
File Network Pattern Run Edit Window Help
Testing
Next Error
Next Prev Save Close Help
F1 Help F4 NewNet F5 SaveNet F6 LoadNet F7 LoadPat. F8 Learn Alt+X Quit

Pattern: 1 Number of Errors: 3
P_Error: 0.0202 Mean Error: 0.14

```
0.86 0.00 0.00
1.00 0.00 0.00
```

Result: (•) Correct ( ) False


NEFCLASS
File Network Pattern Run Edit Window Help
Testing
Next Error
Next Prev Save Close Help
F1 Help F4 NewNet F5 SaveNet F6 LoadNet F7 LoadPat. F8 Learn Alt+X Quit

Pattern: 21 Number of Errors: 3
P_Error: 0.5965 Mean Error: 0.14

```
0.00 0.51 0.42
0.00 0.00 1.00
```

Result: ( ) Correct (•) False

The test results should be saved in a text file for further inspections. By pressing the SAVE button in the dialog box a file dialog will open where you must select a file name, or accept the default name which consists of the name of the network file you have just chosen plus the extension .res. The SAVE commands in the menues file and network won't work here but you will also get a file dialog box. As just described for "saving the new network" you can enter a new name, or choose a result file from the file list.

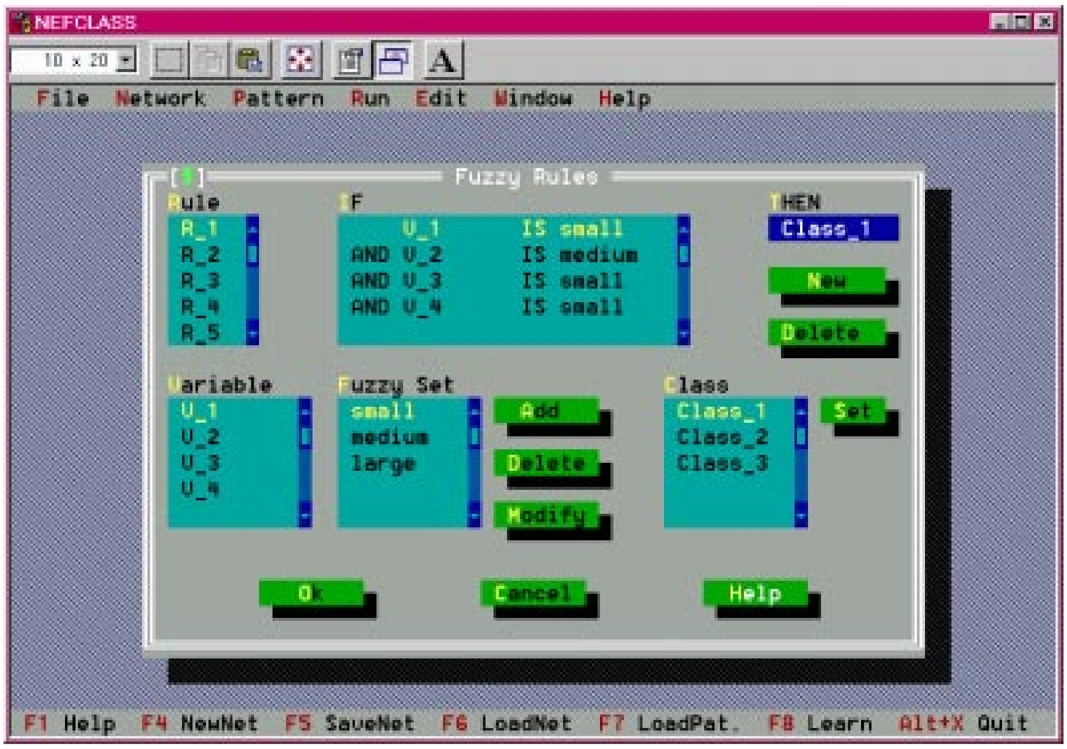
If your network was already saved, or loaded from a file, then the filename with extension ".res" will be offered as default. You can accept this suggestion by simply clicking on OK. Please note that existing files will be overwritten without notification. By default only files with the extension ".res" are displayed in the list of files. After you have pressed the OK button, a message box will inform you about the success of the operation. To view the test result, you can load the saved file into an editor window via the FILE|OPEN command. Now leave the test dialog box by pressing CLOSE.

Test With New Data

Of course it is interesting to see how well the network performs on a pattern set that was not used for training. For continuing the example the second half of the Iris data set should be loaded to see how well the network performs on these new and unknown patterns. Use PATTERN|LOAD (F7), click OK in the message box, select IRIS2.DAT, and click on OPEN. Now open the test dialog RUN|TEST NETWORK again. On this pattern file there are only 2 errors (patterns 27 and 30)! Save also these test results using SAVE, and watch the misclassified patterns outputs by pressing on NEXT ERROR. Leave the box by clicking on CLOSE.

Inspect the Rulebase

To know what rules are embedded within the network open the rule dialog by NETWORK|EDIT RULES. Here you can inspect, but also modify, delete or add rules. In the upper part of the window you see a list with rule names (R_1, R_2, ...), a list containing the antecedent (If part) of the selected rule, and a text area telling you the conclusion (Then part) of the rule.



The first rule of the trained network is

R_1: **IF** V_1 is small
and V_2 is medium
and V_3 is small
and V_4 is small **THEN** Class_1

You can select another rule in the upper left list box by clicking on an entry, using the scrollbar, or by selecting the list box with the TAB key, and selecting an entry with the cursor up and down keys. For now leave the rule editor by clicking on CANCEL or OK. This rule editor will be used again by the demonstration of “building a network by using prior knowledge”.

Now save the rules to a text file by using NETWORK|SAVE RULES. The already known file dialog will open. As the default filename the name of your network file plus the extension .rul will be suggested.

Till now!

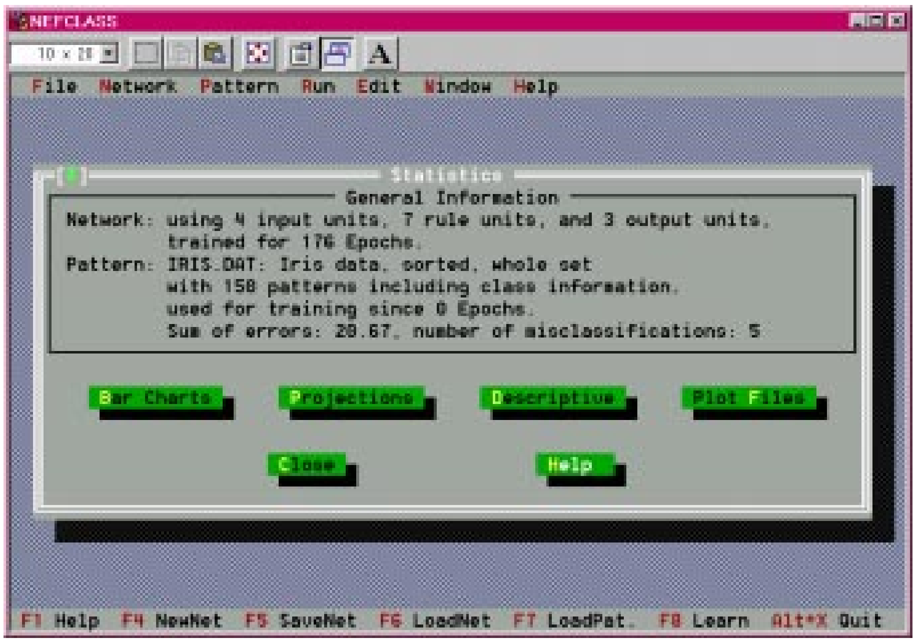
- A network is created with using control and learning parameters. The learning process was viewed at the screen (text and graphic) and the trained network is saved in file with the extension .NET.
- The network is tested with the learnig data and new data. This process was viewed at the screen (text) and the results are saved in files with the extension .RES.
- The rulebase created while the learning process is viewed in a rule editor and saved in a file with the extension .RUL

All these files can be loaded into an editor window, by using the FILE|OPEN (or press F3). command. You will get a file dialog box, where you can select any file and load it (it must be an ASCII file, and less than 64 KB in size). To close the active editor window you can click in the close box in the upper left corner, or press ALT+F3.

5.3.5 Have Some Statistics

In order to get some information about the data it is necessary to run some statistics. These results may help to find some rules which can be put into the rulebase before learning. For this we use the complete Iris data set that is included in the file IRIS.DAT.

Load this file by using the PATTERN|LOAD command (or F7) an be aware that perhaps a network file has to be loaded before, in the case that you have a restart of NEFCLASS-PC. To load means here to load the network and so you won't see the data but a dialog box telling you the success of the execution. Now use the command RUN|STATISTICS. This command will first execute a test with this data set like described on p. 58 and then open a dialog box where you will find some informations about the current network (number of units in every layer and number of epoches the network was trained), the current pattern set (number of patterns, if and how long it was used for learning) and the number of missclassifications in this test.


NEFCLASS
File Network Pattern Run Edit Window Help
Statistics
Bar Charts
Projections
Descriptive
Plot Files
Close
Help
F1 Help F4 NewNet F5 SaveNet F6 LoadNet F7 LoadPat. F8 Learn Alt+X Quit

```
General Information
Network: using 4 input units, 7 rule units, and 3 output units,
         trained for 176 Epochs.
Pattern: IRIS.DAT: Iris data, sorted, whole set
         with 150 patterns including class information.
         used for training since 0 Epochs.
         Sum of errors: 20.67, number of misclassifications: 5
```

By use of the buttons further information can be received.

- **Bar Charts**

  If you click on this button you get a graphical display on the distribution of a single feature on its domain. For each value a bar representing the frequency of this value is drawn. The colors of the bars indicate the class of this value. A red color means that this value can be found in patterns belonging to different classes. Below the bar chart the fuzzy sets for this feature are drawn, and some statistical values are displayed. To view another feature use the cursor up/down keys.

  In this example it can be observed that the features 1 and 2 have a lot of red bars.This means that the feature values vary over almost the whole domain for each class. This fact is responsible for the extreme width of the fuzzy sets. All three fuzzy sets of feature 1 or 2, respectively, almost cover the whole domain. It would be nice, if we could do without these two features. The distribution of the values of the features 3 and 4 is much better, as you can see. By pressing ESC the graphics display will be closed, and the statistics dialog returns.

- **Projections**

  Here you can see a 2-dimensional projection of the feature space. The projection of each pattern is represented by a cross. Different colors indicate different classes (if class information is available). The color coding is explained next to the coordinate system. A pattern that is misclassified has a red circle around its cross (if this information is present). On the outer side of the coordinate system the fuzzy sets of the selected features are displayed. To select other features for the horizontal and vertical axis, use the cursor left/right and up/down keys. Leave the display by pressing ESC.

- **Descriptives**

  By pressing this button you can create a text file where statistical information about the pattern file is presented. After pressing the button the dialog box for saving files appears. By default the pattern filename with extension ".sta" will be suggested. After the file was saved you can view it by loading it into an editor window via the FILE|OPEN command. In this file the following information can be found:

  - General information about the pattern file,
  - for each feature: given and actual ranges, mean, variance, and standard deviation
  - the correlations between the features,
  - the correlation between the features and the class information (if given).

- **Plot Files**

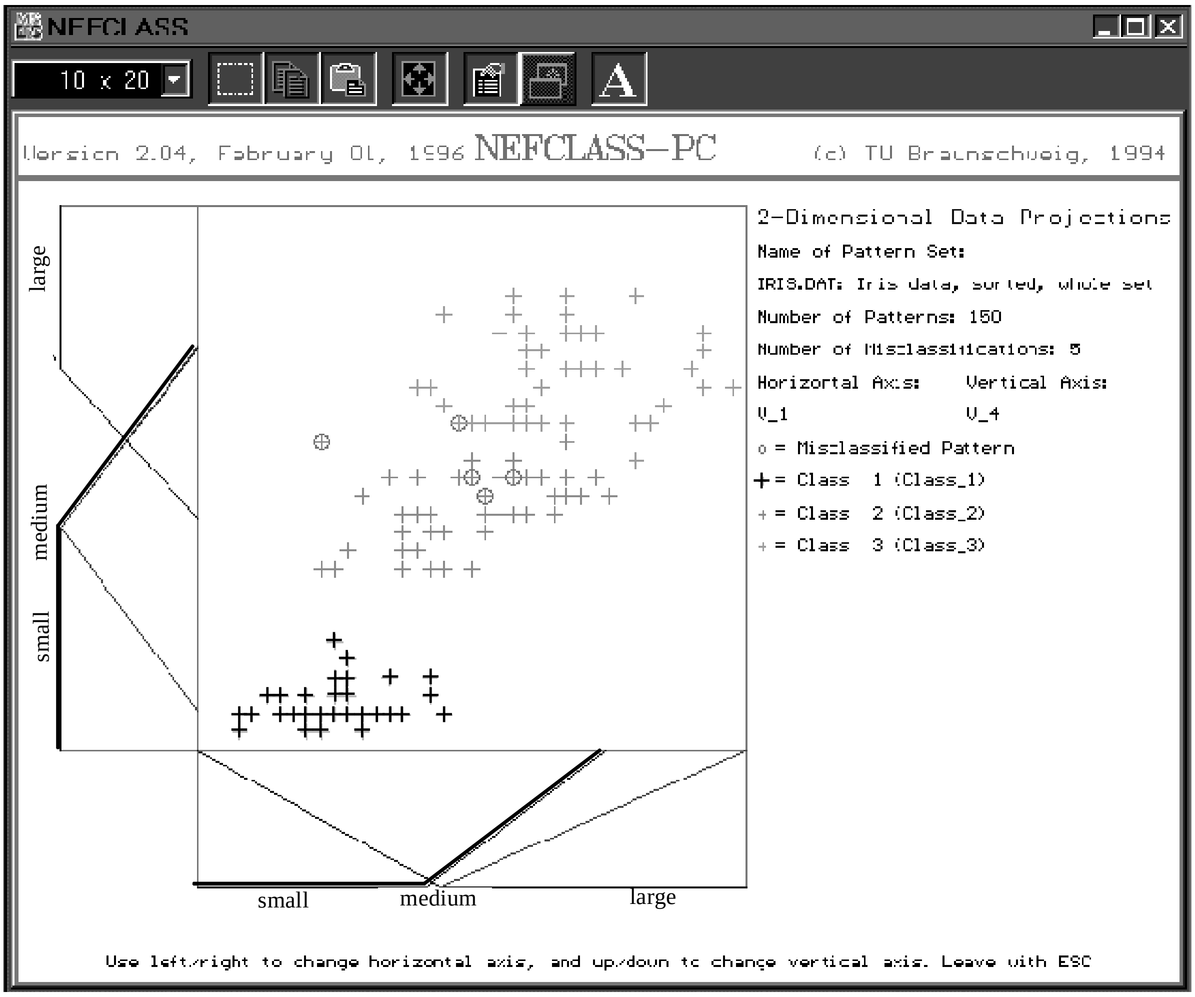
  The bar charts, or the projections can actually be plotted to a file. The file that is created will be in the HPGL format (language of Hewlett Packard plotters). A lot of drawing programs are able to import files of this kind, and so you can use these plots in your own drawings. This feature is only possible, because Ullrich von Bassewitz kindly provides a BGI driver to create a HPGL file from Borland's BGI graphics.

  After you have pressed this button, a dialog box appears where you have to choose whether you want to plot a bar chart or a projection, and which features have to be assigned to the axes. If you select "bar chart", only the selection for axis 1 is valid. If you select "projection", then axis 1 is the horizontal axis, and axis 2 is the vertical axis. After clicking on the OK button, the dialog for saving files appears. By default the pattern filename with extension ".plt" will be suggested. After clicking on OK the plotting begins. Depending on the size of the pattern set, and the speed of your computer this may take some time. A message box will inform you about the success of this operation.

To get some information that can be used as prior knowledge for the rulebase like shown in subsection 5.5 at page 67 click on PROJECTIONS:

You will see a graphical presentation of the patterns in a 2-dimensional projection. Each pattern is represented by a colored cross, where the color denotes the class of the pattern. If a pattern is misclassified a circle is drawn around its cross. On the right to the coordinate system you see a legend. On the horizontal axis feature V_1 and on the vertical axis feature V_2 is displayed. On the outside of the axes the fuzzy sets for these features are drawn.

With the cursor keys (left/right for horizontal axis and up/down for vertical axis) the displayed features can be changed. For this example feature V_3 should be displayed on the horizontal axis, and feature V_4 on the vertical axis like shown in the following screen copy.



Version 2.04, February 01, 1996 NEFCLASS–PC (c) TU Braunschweig, 1994

```
2-Dimensional Data Projections
Name of Pattern Set:
IRIS.DAT: Iris data, sorted, whole set
Number of Patterns: 150
Number of Misclassifications: 5
Horizontal Axis:    Vertical Axis:
V_1                 V_4
o = Misclassified Pattern
+ = Class  1 (Class_1)
+ = Class  2 (Class_2)
+ = Class  3 (Class_3)
```

Use left/right to change horizontal axis, and up/down to change vertical axis. Leave with ESC

You can observe four important aspects in this projection:

- The patterns are distributed in three clusters, where one (class 1) is clearly separable from the remaining two, and the remaining two clusters overlap slightly. In this area you can see the 5 misclassified patterns. This means the Iris problem is obviously easy to solve: **3 rules should be sufficient**, one for each cluster (in fact the Iris problem is very simple and very well known, that's why it is so widely used as a "benchmark" for classification procedures, and that's why we use it here, because the classification result is easy to understand).
- Obviously, **the features 1 and 2 are not necessary to perform the classification** as we hoped while looking at the bar charts. The classes can be seperated in dimensions 3

and 4 alone as seen in the projection.

- The patterns of **class 1 have only "small"** values for the **features 3 and 4**.
- The fuzzy set "medium" for the feature V_4 was shifted as close as possible to the fuzzy set "small". This means, we do not need the fuzzy set "medium", **i.e. 2 fuzzy sets for variable V_4 should be enough** (at least for our current set of rules).

These 4 points should be keept in mind. They will be used later as prior knowledge to initialize a new network (see subsection 5.5, p. 67). Now press ESC to close the graphics, and return to the statistics dialog and press the CLOSE button to leave the statistics dialog.

5.4 Guided Tour through NEFCLASS-PC - *Change a trained Network*

For changing a trained network this second demonstration will use the results of the first one. If the first part has just been completed, you can go on right away. If not (perhaps there was a restart of NEFCLASS-PC) the network file created and saved in the first part has to be loaded by using NETWORK|LOAD (or press F6). If you forgot to save, you can just use the network file DEMO1.NET. Furthermore the pattern file IRIS1.DAT (the data which trained the net) is needed. It must be loaded in any case because the file IRIS.DAT was used at last (use PATTERN|LOAD or press F7).

ATTENTION:

Changing parameters of the network, can have severe consequences to the network structure. If you want to change something later while you have your own experiments be aware of this, and remember to save everything that is possible befor changing a trained network.

This command lets you change almost all parameters of a previously defined network. However, you cannot change the number of input or output units. If there is currently no network defined, a message box will tell you so. The dialog window is the same as for the NETWORK|NEW command.

If you change the maximal number of rule units the network may have, the network has to be trained again. During rule learning the rules that are currently represented in the network will be regarded as a-priori rules. Depending on your changes there will be rules deleted from or added to the network.

If you change the number of fuzzy sets for an input unit, then this unit will receive a new initial partitioning of equally distributed triangular fuzzy sets. This means that for this variable you will lose the current fuzzy sets which may have changed during the learning process.

YOU WILL ALSO LOOSE ALL RULES IN THE NETWORK!

If you change the number of fuzzy sets for any variable, then the rule base uses incorrect fuzzy sets, and therefore the rule base will be deleted.

Changing only the names of the input units (variables) or ouput units (classes) has no effect on the performance or the structure of the network, of course.

The network can also be changed by editing the textfile that can be created with the NETWORK|SAVE command (see also p. 74)

5.4.1 Change Number of Rules

In the first part 7 rules had been used. NETWORK|DISPLAY gives the possibility to check this. In the entry "rule units" you can see the actual number of rules represented within the network, and after this the maximum number of rules the learning algorithm is allowed to create is displayed in brackets.

It would be interesting to know if the network can perform equally well with less rules. In this example a number of 5 rules will be used. Select NETWORK|CHANGE, and change the entry behind "rule units" from 7 to 5, and press OK. Then invoke RUN|LEARNING CONTROL (as desribed at p. 48) select the "Stop, if minimal error ..." option, and enter 50 in the text area below.

Leave with OK, and start to train the network by selecting RUN|START LEARNING (F8). The learning dialog box tells you, that there are 7 prior rules. This are the rules that are already included in the network (since they were learned in part one). Now rule learning starts, and the learning algorithm creates 5 additional rules, resulting in an intermediate rule base of 12 rules. From these 12 rules the best 5 have to be selected. This is done after you have pressed RESUME.

Start training the fuzzy sets by pressing RESUME again. Learning will stop after around 100 epochs, leaving you with only 2 misclassifications! So the network actually performs better with only 5 rules on this part on the Iris data. After learning has stopped, please close the learning dialog box.

Analogous to the example in part one, testing the network with the data used for learning is the next thing to do. RUN|TEST NETWORK (depicted at p. 56, 58) will reveal, that the patterns 60 and 62 are misclassified. When you test your network with the second half of the Iris data from the file IRIS2.DAT you will see that there are now 3 errors (patterns 8, 27 and 30), i.e. one more error than before when you had 7 rules. So the sum of errors for all patterns is again 5, which means the performance on all 150 patterns is the same with only 5 rules. Now save your newly trained network under a new name using NETWORK|SAVE AS.

5.4.2 Change Number of Fuzzy Sets

In the first part we discovered (at p. 64), that obviously only 2 fuzzy sets should be sufficient to partition the domain of the fourth input unit We will check this out now. Be sure that your network is saved because all rules will be deleted by changing the number fuzzy sets. This is necessary, because if you change the number of fuzzy sets for any variable, all the rules that use this variable are now incorrect, because the linguistic values do not exist anymore.

Now invoke NETWORK|CHANGE, select the input unit V_4 in the list, enter 2 into the text field #f-sets, and click on the SET button next to it (don't forget this, or your change will have no effect!). Then leave by clicking on OK.

A message box will pop up, and tell you that all rules have been deleted! So they have to be learned again with learning procedure. Please make sure, that the pattern file IRIS1.DAT is loaded. If you don't know which pattern set is currently loaded you can use the command PATTERN|DISPLAY (like depicted at p. 47). Please also make sure, that learning stops, when the error is not decremented for 50 epochs, and then start the training procedure. An initial rule base with 6 rules will be created, and after the 5 best rules have been selected, you will see that the network classifies only 2 patterns incorrectly after only 10 epochs. Learning will stop after approximately 270 epochs. Testing on the pattern set iris2.dat reveals that the network produces 3 misclassifications in this case. So again we have 5 errors on the complete set.

If you observe the projections of the pattern set, you will see that now the "medium" fuzzy set of V_3 is shifted almost under the fuzzy set "large". Does this mean, we do not need the "medium" fuzzy set? Try this out for your own, by changing the number of fuzzy sets for V_3 to 2 and retrain the network. You should get 2 errors on the training set, but 4 errors on the test set IRIS2.DAT. So the performance decreases a bit.

We have not yet considered the first two features V_1 and V_2. From what we know from the first part is that we should be able to do without them. We will test this in a third part

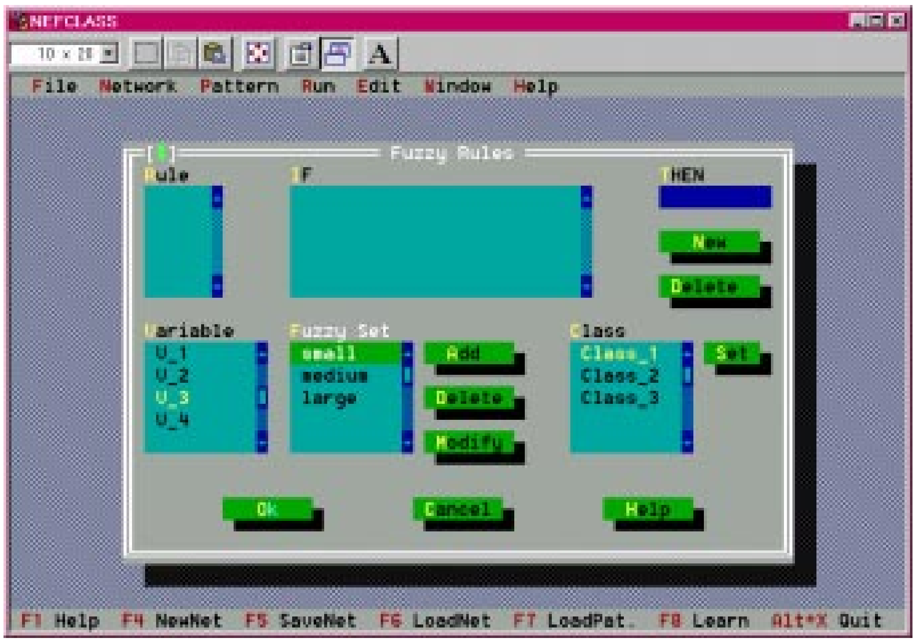
5.5 Guided Tour through NEFCLASS-PC - *Using prior knowledge*

Please create a new network for the Iris problem like depicted in the first part with 4 inputs, 3 outputs, 3 fuzzy sets for each input unit, and this time with 5 rule units. To make sure that each input unit has 3 fuzzy sets, please have a look at the option “same for all units”. The default value while creating a new network is three but if you want to have your own experiments later you can enter the number of fuzzy sets behind it. Then you don't have to enter a value for each unit individually. After this, load the pattern file IRIS1.DAT.

Remember, that we discovered in part one that obviously:

A pattern belongs to **Class 1**, if the values for its **features 3** are **small**
and **features 4** are **small**

We want to use this knowledge to initialize the network, and make it easier for it to learn. Please invoke the rule editor by NETWORK|EDIT RULES.



As you can see there are no rules shown. In part one (p. 58) this comand was invoked after the learning process and then the learned rulebase was shown. But the variables, fuzzy sets and classes are already given because the network was created this way.

In the dialog box click on NEW to create a new, empty rule. Then select

V_3 in the Variable List, and **small** in the Fuzzy Set List,

and click on ADD. Now select

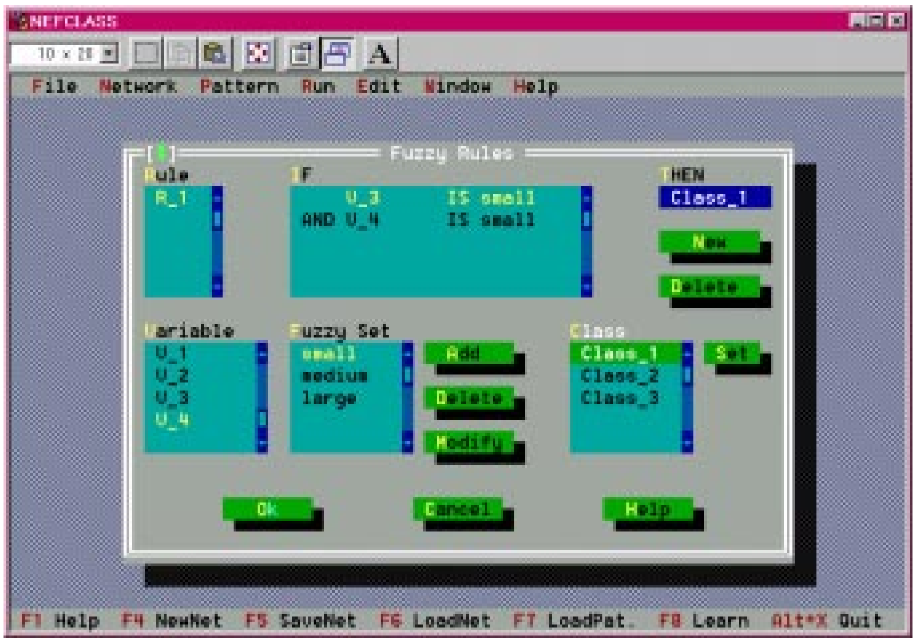
V_4 in the Variable List, and **small** in the Fuzzy Set List,

and again click on ADD. At last select

Class_1 in the Class List

and click on SET.

If you succeeded you should see a screen like the image below



```
R_1
    V_3    IS small
AND V_4    IS small
Class_1
```

Now leave with OK.

You have created the rule:

IF **V_3 is small** and **V_4 is small** then **Class_1**

and included this rule into the network, which has now exactly one rule unit (check this via NETWORK|DISPLAY).

This rule does not use the variables V_1 and V_2 in its antecedent. This kind of rule, that does not use all variables can currently not be learned by the training procedure. If you want to have such a rule, you must enter it manually, or delete some antecedents from a learned rule after training.

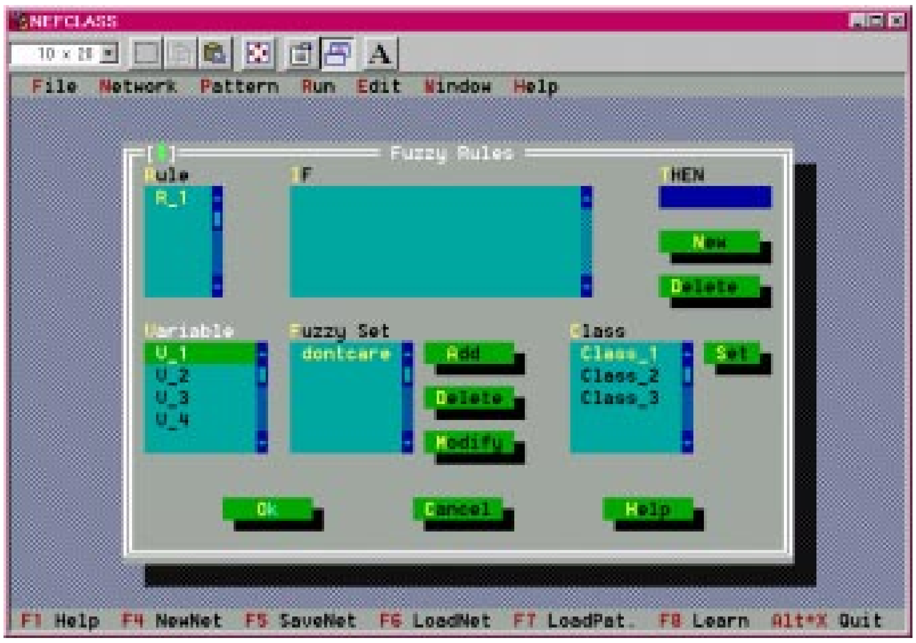
We have now initialized our network with prior knowledge. Let's see how well it learns: Start the learning process, after making sure that learning stops, if the error is not decremented for 50 epochs.

Learning will stop after about 140 epochs, and testing the network with the training set will produce 3 misclassifications (patterns 21, 60, 62). When you use the RUN|TEST NETWORK command on the test set IRIS2.DAT, there will be also 3 errors, however (patterns 8, 27 and 30). So the performance is worse than in part one and two.

Check the rules of the network in the rule editor. You will see that the rule you initialised yourself is the only rule for Class_1. So obviously it is suffient to classify patterns of Class 1. The errors occur only for patterns of Class 2 and 3. When you take a look at the projections (RUN|STATISTICS) of the pattern set, you will see that the partionings of V_3 and V_4 are much nicer now (compare p. 63), i.e. they are more easy to interpret in terms of "small", "medium", and "large", because the fuzzy sets do not overlap so strongly as they did before. On the other hand, the performance is worse now. Please also note, that this time the evolved partitionings do not give us any evidence to assume that 2 fuzzy sets would be enough for V_3!

Another Try

Will we get better, if we use even more prior knowledge? Let's try! Please define a new network, that has only a maximum of 3 rules. Also set the number of fuzzy sets for V_1 and V_2 to 1, and the number of fuzzy sets for V_3 and V_4 to 3. Make sure that the option "individually" in the "Number of Fuzzy Sets" part is selected, and use the SET button for every input. These settings mean that the features V_1 and V_2 are considered as "don't care variables". Because each of them has only one fuzzy set, they will produce a membership value of 1 for any input value. Thus they never produce the minimum in a rule, and so they don't have any influence on the classification result. As you invoke the rule editor you will see that V_1 and V_2 are marked as “don’t cares”.



Now enter the rule:

If **V_3 is small** and **V_4 is small** then **Class_1**.

We have now used the first 3 bits of knowledge that we learned in part one. After the pattern file IRIS1.DAT is loaded, and the control parameters for learning are set to “learning stop, if the error does not decrease for 50 epochs”, the learning process can be started.

The training process creates 4 additional rules, and selects the best 3. The test with the learning data shows only 2 misclassifications(patterns 60 and 62) after 110 epochs (the smallest error occured in epoch 59). Testing the network on IRIS2.DAT we will reveal 3 misclassifications (patterns 8, 27 and 30). A check of the partitionings (RUN|STATISTICS) of V_3 and V_4 , will show, that they look very good, and can easily interpreted in terms of "small", "medium" and "large".

So as a result we can say that the same performance can be reached with only 3 rules using only 2 variables, as using 7 rules and 4 variables.

5.6 Structure of NEFCLASS Files

To use NEFCLASS-PC with your own data you have to create a data file that NEFCLASS-PC is able to work with. The description of how to create such a file is given in the next subsection. Furthermore a NEFCLASS network can be written to or can be read from a text file. A second subsection will describe this.

5.6.1 Pattern Files

To use NEFCLASS-PC on your own data you have to write an ASCII data file. The following types of lines can be written:

- Comment lines beginning with a %.

  You can (but need not) write them everywhere in the file,

  - but not within the pattern data section, and
  - not in a line directly following a command.
- Command lines containing only a command. In the following line the parameters belonging to the command are written. The commands are:
  - NAME — Name you want to give to your pattern file. This name can have up to 60 arbitrary characters.
  - INRANGES — The ranges for each input feature should be specified. If not, the exact ranges will be computed during the patterns are loaded. By using the INRANGES command you can specify larger ranges as really are needed for the patterns.
  - PATTERNS — the number of patterns, the number of input features, and the number of classes (outputs) has to be specified in one line.

    After this start to write the patterns following these instructions:

    - seperate the feature or class values by whitespace (blank, tab, newline)
    - the input features may be real or integer values
    - the classes must be coded as binary vectors. For example, if there are 3 classes, class 2 will be encoded as 0 1 0.
  - UNKNOWN — If you want to use a pattern file, without classification information, then the command UNKNOWN must be used before the PATTERNS command. In this case you will only specify the number of patterns, and the number of input features in the line after the PATTERNS command. Obviously, you cannot use such a pattern file for training

a network, but you can use it to obtain the classification information for your patterns from a previously trained network.

- END — The END command must be written on a single line after the patterns. After this command the pattern file will not be further processed.

As a first example see the beginning and end of IRIS1.DAT we used for the guided tour. It is a file for training a network, and therefore it has classification information.

```
% This is the first half of the IRIS data set reformatted
% for use in NEFCLASS-PC 2.0
% Class 1 0 0 is Iris-setosa, class 0 1 0 is Iris-versicolor, and
% class 0 0 1 is Iris-virginica. The classes are interleaved.
% This file contains a data set with 75 cases, 25 for each class

% Name of the pattern set
NAME
IRIS1.DAT: Iris data, first part

% Ranges of the 4 input variables
INRANGES
4 8
2 5
1 7
0 3

% This are 75 patterns,
% first given are the number of patterns, inputs and outputs
PATTERNS
75 4 3
5.1  3.5  1.4  0.2  1 0 0
...
6.7  3.3  5.7  2.1  0 0 1
END
```

Furthermore see below a formalized definition how to write a pattern file:

Format:

```
[  <comment>   ]  <data area>   ... END
   <comment>:   lines beginning with %
   <data area>: <name> | <inranges> | <unknown> | <pattern>
                <name>:       NAME <newline> <string of max. 60 characters>
                <inranges>:   INRANGES <newline><min max> ... <newline>
                <unknown>:    UNKNOWN <newline>
                <pattern>:    PATTERNS <newline>
                              <n. of patterns> <n. of inputs> <n. of outputs> <newline>
                              <in feature> ... <out feature>... <newline>
   END
```

As a second example see the beginning and end of the file IRISNOCL.DAT that is also distributed together with NEFCLASS-PC. It doesn't contain classification data

```
% This is the IRIS data set reformatted for use in
% NEFCLASS-PC 2.0 There is no class information within
% the data, but the patterns are sorted by their class
% Class 1 is Iris-setosa (1-50), class 2 is Iris-versicolor
% (51-100), and class 3 is Iris-virginica (101-150).This file
% contains the whole data set with 150 cases, 50 for each class

% Name of the pattern set
NAME
IRISNOCL.DAT: Iris data, sorted, no class information

% There is no class information
UNKNOWN

% Ranges of the 4 input variables
INRANGES
4 8
2 5
1 7
0 3

% This are the 150 patterns,
% first given are the number of patterns, inputs and outputs
% (here: 0)
PATTERNS
150 4 0
5.1  3.5  1.4  0.2
...
5.9  3.0  5.1  1.8

END
```

5.6.2 The Network File

This is the description of the file format of a trained network. Usually you won't write a network file on your own, because you can create networks more conveniently by using the NETWORK|NEW command, and save it to a file by using NETWORK|SAVE. But you maybe want to edit a network file, to change some specific parameters. As an example see the file DEMO1.NET created in the first part of the guided tour.

You can write comment lines beginning with a % everywhere in the file, but not within the network data, and not in a line directly following a command.

```
% NEFCLASS network file created by NEFCLASS-PC 2.0, (c) TU Braunschweig, 1994, 1995
% Filename: E:\NEFCLASS\DEMO1.NET
% This file was created on 10/8/1995 at 19:55.
```

The network structure is defined by the PARAMETERS command. After this command

- the number of input units (L1 <= 50),
- rule (hidden)units (L2 <= 500),
- output units (L3 <= 50),
- and the maximum number of rule units that can be created during learning (max_rules <= 500) are specified.

These four integer values are followed by a number of values that specify

- the number of fuzzy sets for each of the L1 input units. If an input unit has only one fuzzy set, it means it is ignored, because the fuzzy set has a membership value of 1 for the whole domain of the respective input variable (don't care variable).

The last entries in the PARAMETERS section define the domains of the input units. For each of the L1 input units a pair of values must be specified defining the lower and upper bounds of the input range for the respective unit. All values of this section are seperated by an arbitrary amount of whitespace (blank, tab, newline).

```
% These are the network parameters
PARAMETERS
        4               7               3               7

        3               3               3               3

        4.0000          8.0000          2.0000          5.0000

        1.0000          7.0000          0.0000          3.0000
```

After the parameters the commands USESUM or USEMAXIMUM, respectively, (to define the output function), and INDIVIDUALLY or SAME, respectively, (to specify whether each input variable has an individual number of fuzzy sets or not) are used.

```
USESUM
INDIVIDUAL
```

After the command VNAMES you can specify a name for each input and each output unit (up to 8 characters). Each name has to be written on a single line, and there must be no blank lines or comments within the name list.

```
% This are the names of the input and output units
VNAMES
V_1
V_2
V_3
V_4
Class_1
Class_2
Class_3
```

After the command FUZZY the parameters of the fuzzy sets are defined. If the fuzzy sets have names, the command NAMES must appear before the command FUZZY. A fuzzy set is described by five parameters. The first three values specify the left spread, center, and right spread of a triangular membership function. The last two values are either 0 or 1. If the first of these two values is 1, then the triangle is left shouldered, and if the second value is 1, then the triangle is right shouldered. If the fuzzy sets have names, then the respective name appears right after the last parameter of each fuzzy set (separated by a blank).

```
% This are the parameters of the fuzzy sets
NAMES
FUZZY
4.0000000000   5.6248726679   6.9177715904   1  0  small
4.0001299074   5.9691544158   7.9999606291   0  0  medium
4.0089878883   5.9691544158   8.0000000000   0  1  large
2.0000000000   3.9627328050   4.9999998437   1  0  small
2.0008956379   3.9627328050   4.9999999870   0  0  medium
3.5000000000   4.2500000000   5.0000000000   0  1  large
1.0000000000   2.2038804727   3.5518469899   1  0  small
1.8248094912   4.0317061825   6.2143308622   0  0  medium
3.7547419340   5.3360951920   7.0000000000   0  1  large
0.0000000000   1.2422995140   2.2285130517   1  0  small
0.3214891145   1.2447540273   2.2322922980   0  0  medium
1.2663533515   2.0933809492   3.0000000000   0  1  large
```

The network structure if defined after the MATRIX command. For each input unit there is one line, and for each rule unit there is one row in the antecedent matrix. Each entry specifies an index that refers to a fuzzy set of the respective input unit. If an entry is 0, this means that there is no connection from the respective input unit to the respective rule unit (i.e. the represented fuzzy rule does not use this specific variable in its antecedent).

```
% This is the connection matrix
MATRIX
    1  1  2  3  2  3  2
    2  1  1  1  1  2  1
    1  1  3  3  2  3  2
    1  1  3  3  2  3  1
```

Each rule unit is connected to exactly one output unit. After the WEIGHTS command for each rule unit there is a pair of values specified. The first value is the number of the output unit (a value from [1, L3]), and the second value is the weight that connects the rule unit to this output unit.

```
% This are the conclusion weights
WEIGHTS
    1   1.0000000000
    1   1.0000000000
    3   1.0000000000
    3   1.0000000000
    2   1.0000000000
    3   1.0000000000
    2   1.0000000000
```

The network specification is completed by the "END" command. After this command the file will not be further processed.

```
END
```

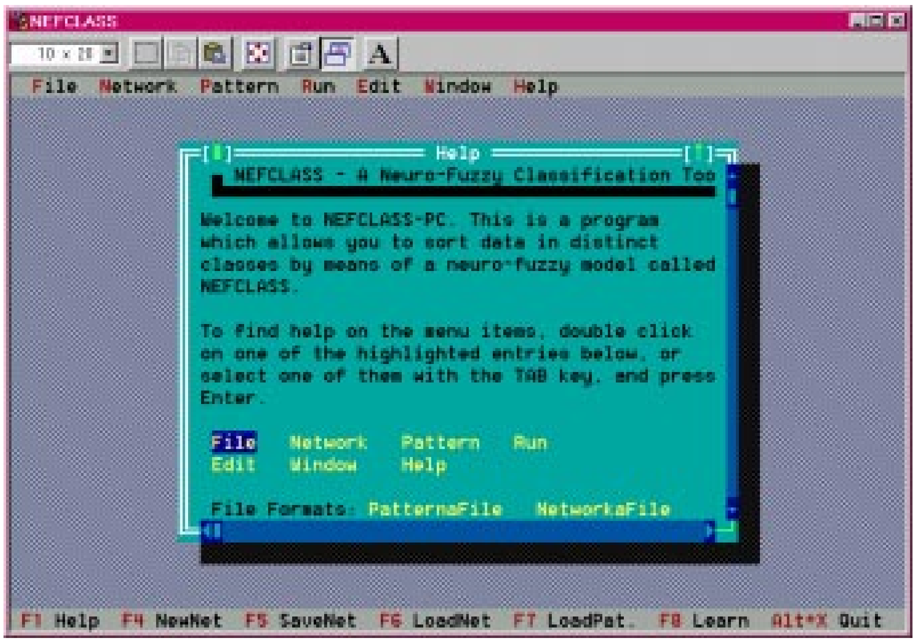
Below you will find the format in a condensed description:

Format:

```
[   <comment>    ]   <parameters>  [[  <comment>    ]   <data area>]   ...  END
    <comment>:        lines beginning with %
    <parameters>:     PARAMETERS <newline>
                          <L1> <L2> <L3> <max_rules> <AntecBase.count array>
                          <Ranges of input variables: array of <min max> pairs>
                          <newline>
                      [USESUM <newline> | USEMAXIMUM <newline>]
                      [INDIVIDUAL <newline> | SAME <newline>]
```

	[VNAMES <newline> <var_names>]
	<var_names>: <<string> <newline> <string> <newline> ...>
<data area>:	<fuzzy sets> | <connections> | <weights>
	<fuzzy sets>: [NAMES <newline>] FUZZY <newline>
	<<a b c ls rs [n]> <a b c ls rs [n]> ...> <newline>
	<connections>: MATRIX <newline>
	<Antecedents array> <newline>
	<weights>: WEIGHTS <newline>
	<ConclusionWeights array> <newline>
<a b c ls rs [n]>:	a = left point, b = peak, c = right point,
	ls = 1 if left shouldered, rs = 1 if right shouldered), n is an optional name (will be only read if command NAMES is specified before FUZZY)

While working with NEFCLASS-PC you also can get all these instructions by online help. Using the HELP|INDEX command a dialog box opens where you can choose the subject you are interested in.



NEFCLASS - A Neuro-Fuzzy Classification Too

Welcome to NEFCLASS-PC. This is a program
which allows you to sort data in distinct
classes by means of a neuro-fuzzy model called
NEFCLASS.

To find help on the menu items, double click
on one of the highlighted entries below, or
select one of them with the TAB key, and press
Enter.

File Network Pattern Run
Edit Window Help

File Formats: PatternsFile NetworksFile

F1 Help F4 NewNet F5 SaveNet F6 LoadNet F7 LoadPat. F8 Learn Alt+X Quit

If you have just opened a window, e.g. by using RUN|LEARNING CONTROL (see p. 48) you can click the HELP button and you will get context sensitive help.