

# EDL-Einführung

N2E4 und sein Gebrauch

## N2E4

- N2E4 erlaubt protokollierte Experimente mit neuronalen Netzwerken auf beliebigen Daten vorzunehmen.
- Die Experimente und deren Protokolle Ausgaben werden in EDL-Skripten festgelegt.
- Eine Dokumentation findet sich unter

<http://www.neuro.informatik.uni-kassel.de/projekte/n2e4/n2e4.htm>

# Verzeichnisbaum



## Pfade

- Damitedl-Skriptenausgeführtwerden können `edl <Skriptname>.edl`,müssendie PfadedesKommando-Fenstersgesetztsein

- `path = p:\n2e4\bin\win32\`

BeiDOS/Windowsundsonst

- `path = p:/n2e4/bin/aix/` oder
- `path = p:/n2e4/bin/linux/`

# Datenfiles

- Die Datensollten in Ascii-files vorliegen, in denen sie spaltenweise (durch <tab> oder <space> getrennt) aufgelistet sind.
- Die erste Spalte enthält eine laufende Nummer, die auch als Zeitrealisiert sein kann.
- Die einzelnen Spalten( **Kanäle**) erhalten in der ersten Zeile jeweils einen Kanal-Namen( **time...**).
- textuelle Werte müssen in numerische Werte umgewandelt werden.
- Am Ende darf keine Leerzeile stehen!!!

## textfile

| x    | f(x)         | Farbe | //in numerische Wertewandeln   |
|------|--------------|-------|--------------------------------|
| -1   | 1,02764955   | blau  | //,in Dezimalpunkt wandeln     |
| -0,8 | -0,337542675 | grün  | //whitespaces vor/nl entfernen |
| -0,6 | -0,983635455 | blau  | //xin Time umbenennen          |
| -0,4 | -0,91780001  | gelb  |                                |
| -0,2 | -0,518454747 | rot   |                                |
| 0    | 0,009470245  | rot   |                                |
| 0,2  | 0,601720314  | blau  |                                |
| 0,4  | 0,866552841  | grün  |                                |
| 0,6  | 0,94885819   | gelb  |                                |
| 0,8  | 0,336255228  | rot   |                                |
| 1    | -0,99858039  | gelb  |                                |

| /whitespaces und /nl am Dateiende entfernen!!

# textfilebearbeiten:

- Wenn der Datenfile keine neue Zählerpalte erhalten soll, muß die erste Spalte in **Time**(nicht time) umbenannt werden.
- Dezimale Kommatamüssen in Dezimalpunkte umgewandelt werden.
- Zwischen den Daten darf jeweils nur ein TAB oder SPACE stehen
- Vor dem Zeilenende (NL) müssen whitespaces entfernt werden.
- Symbolische Werte müssen in numerische Werte (abzählen) umgewandelt werden.
- Alle whitespaces (incl. NL) am Dateiende müssen entfernt werden.

Übersetzen mit:

```
txt2rdf -i DatenB.txt -o Daten.rdf -l NN-Uebung -t Beispiel -D 1.0
```

oder (falls die erste Spalte **Time** heißt):

```
txt2rdf -i DatenB.txt -o Daten.rdf -l NN-Uebung -t Beispiel
```

# textfilebearbeitet

| x    | f(x)         | Farbe |
|------|--------------|-------|
| -1   | 1.02764955   | 1     |
| -0.8 | -0.337542675 | 2     |
| -0.6 | -0.983635455 | 1     |
| -0.4 | -0.91780001  | 3     |
| -0.2 | -0.518454747 | 4     |
| 0    | 0.009470245  | 4     |
| 0.2  | 0.601720314  | 1     |
| 0.4  | 0.866552841  | 2     |
| 0.6  | 0.94885819   | 3     |
| 0.8  | 0.336255228  | 4     |
| 1    | -0.99858039  | 3     |

# Der Gebrauch von txt2rdf

Usage: txt2rdf: [ option ]

## Options:

-----

```
-i<name>      : input file name (default is no input file => stdin)
-o<name>      : output file name (default is output.rdf)
-O<name>      : output file name generator file
-b<bool>      : begin new file if time time warp occurs (needs -O)
               : (default is n)
-l<name>      : laboratory name (default is FGNN internal)
-t<name>      : data type description (default is <not specified>)
-d<date>      : sampling date (default is today)
-D<number>    : time step between samples (0 (default) => non-equidistant)
-m<bool>      : write mixed statistical moments (default y)
-h            : print this help

-s<name> <value> : adds the symbol <name> with the value <value>
                  : to the [Description] tag. Can be used multiple times
```

# eine batch-datei für DOS

```
@echo off
```

```
rem keine Leerzeichen vor und nach dem ==-Zeichen !!!
```

```
set rohdaten="beispiel.txt"
```

```
set rdffdaten="beispiel.rdf"
```

```
set lab="NN_Uebung"
```

```
set dattype="Funktionswerte"
```

```
txt2rdf -i%rohdaten% -o%rdffdaten% -l%lab% -t%dattype%
```

- ersetzt den Aufruf:

```
txt2rdf -ibeispiel.txt -obeispiel.rdf -lNN_Uebung -tFunktionswerte
```

# rdf-Format

|                     |                         |
|---------------------|-------------------------|
| [Description]       | [Channels] //Kanalnamen |
| Laboratory ...      | [Data] //Rohdaten       |
| OriginalFilename... | [Statistics] //proKanal |
| Translator ...      | (AnzahlDerSamples)      |
| TranslationDate ... | (SummeallerWerte)       |
| NumberOfChannels... | (SummederQuadrate)      |
| DataType ...        | (Minimum)               |
| DeltaT ...          | (Maximum)               |
| SamplingDate ...    | (?mixedmoments?)        |
|                     | [(Korrelationen)]       |

# rdf-Kopf

```
[Description]
Laboratory  NN_Uebung
OriginalFilename  DatenB.txt
Translator  <<UNKNOWN>>
TranslationDate  15.05.2000 14:32:35
NumberOfChannels  3
DataType  Beispiel
DeltaT  1
SamplingDate15.05.2000 14:32:35

[Channels]
Time      x      f(x)  Farbe
```

# rdf-Daten

[Data]

|    |                  |              |            |   |
|----|------------------|--------------|------------|---|
|    | 4.262997402e-314 | -1           | 1.02764955 | 1 |
| 1  | -0.8             | -0.337542675 | 2          |   |
| 2  | -0.6             | -0.983635455 | 1          |   |
| 3  | -0.4             | -0.91780001  | 3          |   |
| 4  | -0.2             | -0.518454747 | 4          |   |
| 5  | 0                | 0.009470245  | 4          |   |
| 6  | 0.2              | 0.601720314  | 1          |   |
| 7  | 0.4              | 0.866552841  | 2          |   |
| 8  | 0.6              | 0.94885819   | 3          |   |
| 9  | 0.8              | 0.336255228  | 4          |   |
| 10 | 1                | -0.99858039  | 3          |   |

//beachte die et was merkwürdige 0 (am besten ersetzen)

# rdf-Daten mit Time

[Channels]

| Time | f(x) | Farbe |
|------|------|-------|
|------|------|-------|

[Data]

|      |              |   |
|------|--------------|---|
| -1   | 1.02764955   | 1 |
| -0.8 | -0.337542675 | 2 |
| -0.6 | -0.983635455 | 1 |
| -0.4 | -0.91780001  | 3 |
| -0.2 | -0.518454747 | 4 |
| 0    | 0.009470245  | 4 |
| 0.2  | 0.601720314  | 1 |
| 0.4  | 0.866552841  | 2 |
| 0.6  | 0.94885819   | 3 |
| 0.8  | 0.336255228  | 4 |
| 1    | -0.99858039  | 3 |

# rdf-Statistics

```
[Statistics]
  11          11          11          11
  55          2.220446049e-16  0.034493091  28
 385          4.4          6.372322633  86
 4.262997402e-314      -1      -0.99858039  1
 10           1          1.02764955  4
true
  22
 3.222869065  0.610080722
159   3.8   -1.889728985
```

## Vorverarbeitung

- Bevor die Daten dem Netzwerk zum Training vorgeführt werden können, müssen sie vorbereitet werden:
- Skalierung
- Teilung in Trainings- und Testdaten
- ...



# Select.edl

```
{ Auswählen der Kanäle; hier x, Farbe und f(x) (vgl. RDF-Datei) }
PROGRAM LEVEL 1.3 { Diese erste Zeile ist obligatorisch }
( Titel      = "Beispiel",
  Keywords   = "Wendeparabel, Beispiel, MLP, Backprop",
  Rem1       = "Beispiel-EDL-Skript für NN-Übung",
  Rem2       = "Wendeparabel mit MLP lernen",
  Rem3       = "Hier: [-1,1] auf Wendeparabel abbilden",
  Rem4       = "Datenvorverarbeitung Select");
SECTION StdResult (BOOLEAN Error; STRING ErrorMessage);
BEGIN CALL "Select" ( Input      = "Daten.rdf",
                     Output     = "selected.idf",
                     Channels    = ["x","Farbe","f(x)"],
                     Epsilon    = 0.001 )
  GIVING StdResult; END.
```

# select.prt

```
[EDL-PROTOCOL]
EDL-Interpreter-Version      $Revision$      EDL-Level
1.4                          Mar 22 2000     14:02:12
Operating-System             WINDOWS32
Compiler                     Gnu Gcc
EDL-Program                  select
ExecutionDate 16.05.2000 13:49:50
ExecutionHost PC-NN15
Titel                        Beispiel
Keywords                     Wendeparabel, Beispiel, MLP, Backprop
Rem1 Beispiel-EDL-Skript für NN-Übung
Rem2 Wendeparabel mit MLP lernen
Rem3 Hier: [-1,1] auf Wendeparabel abbilden
Rem4 Datenvorverarbeitung Select

[EDL-CALL]
Calling                      Select
Start-Time                   16.05.2000 13:49:51
Success
End-Time                     16.05.2000 13:49:52

[Select]
```

# selected.idf

```
[Description]
  Laboratory      FG Neuronale Netzwerke
  OriginalFilename daten.rdf
  Translator      <<UNKNOWN>>
  TranslationDate 16.05.2000 13:49:52
  NumberOfChannels 3
  DataType        Incremental Data File (IDF)
  DeltaT          1
  SamplingDate    15.05.2000 14:32:35
[Channels]
  Time x          Farbe    f(x)                //Reihenfolge verändert
[Data]
  0      -1        1        1.02764955
  1      -0.8      2        -0.337542675
  2      -0.6      1        -0.983635455
  3      -0.4      3        -0.91780001
  4      -0.2      4        -0.518454747
  5       0        4        0.009470245
  6      0.2       1        0.601720314
  7      0.4       2        0.866552841
  8      0.6       3        0.94885819
  9      0.8       4        0.336255228
  10     1         3        -0.99858039
[Statistics]...
```

# scaleIn.edf

```
{ Skalieren der Input-Kanäle; hier x und Farbe (vgl. RDF-Datei) }
PROGRAM LEVEL 1.3 { Diese erste Zeile ist obligatorisch }
( Titel      = "Beispiel", ... );
SECTION ScaledResult (BOOLEAN Error; STRING ErrorMessage;
                     NUMBER NumOfSamples;; { SECTION ScaledResult }

BEGIN
CALL "Scale" (
    Input      = "selected.idf",
    Output     = "scaled.idf",
    Channels   = ["x","Farbe"],
    Method     = "Linear",
    Base       = "Internal",
    Domain     = "Mean",
    Range      = [-1,1],
    KeepInput  = TRUE,
    Epsilon    = 0.001
)
GIVING ScaledResult;

END.
```

# scaled.idf

```
[Description]
Laboratory      FG Neuronale Netzwerke
OriginalFilename selected.idf
Translator      <<UNKNOWN>>
TranslationDate 18.05.2000 08:38:9
NumberOfChannels 3
DataType        Incremental Data File (IDF)
DeltaT          1
SamplingDate    15.05.2000 14:32:35
[Channels]
Time x          Farbe      f(x)
[Data]
0      -1.58113883      -1.335646142      1.02764955
1      -1.264911064      -0.4714045208      -0.337542675
2      -0.9486832981      -1.335646142      -0.983635455
3      -0.632455532      0.3928371007      -0.91780001
4      -0.316227766      1.257078722      -0.518454747
5      0                  1.257078722      0.009470245
6      0.316227766      -1.335646142      0.601720314
7      0.632455532      -0.4714045208      0.866552841
8      0.9486832981      0.3928371007      0.94885819
9      1.264911064      1.257078722      0.336255228
10     1.58113883      0.3928371007      -0.99858039
[Statistics]...
```

# scaleOut.edl

```
{ Skalieren des Output-Kanals, hier f(x), auf das Intervall [0.05,0.95]}
PROGRAM LEVEL 1.3 { Diese erste Zeile ist obligatorisch }
( Titel      = "Beispiel",
  Keywords   = "Wendeparabel, Beispiel, MLP, Backprop",
  Rem1       = "Beispiel-EDL-Skript für NN-Übung",
  Rem2       = "Wendeparabel mit MLP lernen",
  Rem3       = "Hier: [-1,1] auf Wendeparabel abbilden",
  Rem4       = "Datenvorverarbeitung Scale input");
SECTION ScaledResult ( BOOLEAN Error;STRING ErrorMessage; NUMBER NumOfSamples;;
BEGIN
    CALL "Scale" (
        Input      = "scaled.idf",
        Output     = "temp.idf",
        Channels    = ["f(x)"],
        Method      = "Linear",
        Base        = "Internal",
        Domain      = "Max",
        Range       = [0.05,0.95],
        KeepInput   = TRUE,
        Epsilon     = 0.001
    )
    GIVING ScaledResult;
END.
```

# temp.idf

```
[Description]
  Laboratory      FG Neuronale Netzwerke
  OriginalFilename scaled.idf
  Translator      <<UNKNOWN>>
  TranslationDate 18.05.2000 10:43:59
  NumberOfChannels 3
  DataType        Incremental Data File (IDF)
  DeltaT1
  SamplingDate    15.05.2000 14:32:35
[Channels]
  Time  x          Farbe      f(x)
[Data]
  0      -1.58113883      -1.335646142      0.95
  1      -1.264911064      -0.4714045208      0.343616204
  2      -0.9486832981      -1.335646142      0.05663816146
  3      -0.632455532      0.3928371007      0.08588059803
  4      -0.316227766      1.257078722      0.2632596455
  5      0                  1.257078722      0.4977505507
  6      0.316227766      -1.335646142      0.7608130253
  7      0.632455532      -0.4714045208      0.8784449236
  8      0.9486832981      0.3928371007      0.9150028743
  9      1.264911064      1.257078722      0.6429001603
  10     1.58113883      0.3928371007      0.05
[Statistics]...
```

## 1. separate.edl

```
PROGRAM LEVEL 1.3 { Auswählen der Test- und Trainingssamples }
( Titel      = "Beispiel", ...);
{ Hilfsvariablen }
  NUMBER START_IDX          = 0;
  NUMBER LESSONSAMPLE_STEP  = 3; { 3 Samples lernen }
  NUMBER TESTSAMPLE_STEP    = 1; { 1 Sample testen }
  STRING Kommentar;
  NUMBER i=1;
  NUMBER Samplenr=1;
  NUMBER LIST_END=1;
  NUMBER NumLessonSamples=0;
  NUMBER NumTestSamples=0;
  BOOLEAN IsLessonSample = TRUE;
  NUMBER Switchnr;
  NUMBER NumOfSamples = 11;
  LIST OF NUMBER LessonSampleSet = [ 1 ]; { wird hier initialisiert }
  LIST OF NUMBER TestSampleSet   = [ 2 ]; { dto. }

SECTION StdResult ( BOOLEAN Error; STRING ErrorMessage); { SECTION StdResult }
```

## 2. separate.edl

```
BEGIN {Aufteilung der Listen von Indizes der Samples in Trainings- und Testdaten:}
  Samplenr=START_IDX;
  Switchnr = Samplenr + LESSONSAMPLE_STEP - 1; {ist schon richtig so...}
  IsLessonSample = TRUE;
  WRITE(Switchnr);
  WHILE (Samplenr<=NumOfSamples)
  DO BEGIN
    IF (IsLessonSample==TRUE)
    THEN BEGIN
      NumLessonSamples=NumLessonSamples+1;{ Resize Liste: }
      LessonSampleSet = LessonSampleSet[1, NumLessonSamples];
      LessonSampleSet[NumLessonSamples,1] = [Samplenr];END
    ELSE BEGIN
      NumTestSamples=NumTestSamples+1;
      TestSampleSet = TestSampleSet[1, NumTestSamples];
      TestSampleSet[NumTestSamples,1] = [Samplenr]; END;
    IF (Samplenr>=Switchnr)
    THEN BEGIN
      IF (IsLessonSample)
      THEN BEGIN
        Switchnr=Switchnr+TESTSAMPLE_STEP; IsLessonSample=FALSE; END
      ELSE BEGIN
        Switchnr=Switchnr+LESSONSAMPLE_STEP; IsLessonSample=TRUE; END
      END;
      Samplenr=Samplenr + 1;
    END;
  END;
```

## 3. separate.edl

```
{ Aufteilung der Daten in Lektion und Testset }
{ Trainingssamples }
CALL "Select" ( Input      = "temp.idf",
                Output     = "lesson.idf",
                Channels    = ["x","f(x)"],
                SampleSet   = LessonSampleSet,
                KeepInput   = TRUE
              )
GIVING StdResult;

{ Testsamples }
CALL "Select" ( Input      = "temp.idf",
                Output     = "test.idf",
                Channels    = ["x","f(x)"],
                SampleSet   = TestSampleSet,
                KeepInput   = TRUE
              )
GIVING StdResult;

END.
```

# lesson.idf

```
[Description]
  Laboratory      FG Neuronale Netzwerke
  OriginalFilename temp.idf
  Translator      <<UNKNOWN>>
  TranslationDate 18.05.2000 11:06:53
  NumberOfChannels 2
  DataType        Incremental Data File (IDF)
  DeltaT0
  SamplingDate    15.05.2000 14:32:35
[Channels]
  Time  x          f(x)
[Data]
  0      -1.58113883      0.95
  → 1      -1.264911064    0.343616204
  3      -0.632455532    0.08588059803
  4      -0.316227766    0.2632596455
  → 5      0              0.4977505507
  7      0.632455532    0.8784449236
  8      0.9486832981    0.9150028743
  → 9      1.264911064    0.6429001603
[Statistics]...
```

# test.idf

```
[Description]
  Laboratory      FG Neuronale Netzwerke
  OriginalFilename temp.idf
  Translator      <<UNKNOWN>>
  TranslationDate 18.05.2000 11:06:54
  NumberOfChannels 2
  DataType        Incremental Data File (IDF)
  DeltaT0
  SamplingDate    15.05.2000 14:32:37
[Channels]
  Time  x          f(x)
[Data]
  2      -0.9486832981    0.05663816146
  6      0.316227766    0.7608130253
  10     1.58113883      0.05
[Statistics]...
```

# lesson.edl

```
PROGRAM LEVEL 1.3 { erstellen der Patern-Dateien }
( Titel = ...);
SECTION ScaledResult ( BOOLEAN Error;STRING ErrorMessage; NUMBER NumOfSamples;);
BEGIN
{ Aufbau der Lektion }
    CALL "Mklesson"
    ( Input      = "Lesson.idf",
      Target     = ["f(x)"],
      KeepInput  = TRUE,
      Output     = "Learn.PAT" )
    GIVING StdResult;

    CALL "Mklesson"
    ( Input      = "Test.idf",
      Target     = ["f(x)"],
      KeepInput  = TRUE,
      Output     = "Test.PAT" )
    GIVING StdResult;
END. { PROGRAM }
```

# Lektion.pat

```
[Header]
PatternRepresentation      TargetMode
StartTime      18.05.2000 12:25:25
EndTime        18.05.2000 12:25:34
DimOfPatterns  1          1
DimOfTargets   1          1
DimOfInfos     0
NumOfDataSets  8

[Data.Channels]
Number Time      x          f(x)

[Data]
0      0      -1.58113883      0.95
1      1      -1.264911064      0.343616204
2      3      -0.632455532      0.08588059803
3      4      -0.316227766      0.2632596455
4      5      0                0.4977505507
5      7      0.632455532      0.8784449236
6      8      0.9486832981      0.9150028743
7      9      1.264911064      0.6429001603
```

# Test.pat

[Header]

|                       |            |            |
|-----------------------|------------|------------|
| PatternRepresentation |            | TargetMode |
| StartTime             | 18.05.2000 | 12:25:27   |
| EndTime               | 18.05.2000 | 12:25:37   |
| DimOfPatterns         | 1          | 1          |
| DimOfTargets          | 1          | 1          |
| DimOfInfos            | 0          |            |
| NumOfDataSets         | 3          |            |

[Data.Channels]

|        |      |   |      |
|--------|------|---|------|
| Number | Time | x | f(x) |
|--------|------|---|------|

[Data]

|   |    |               |               |
|---|----|---------------|---------------|
| 0 | 2  | -0.9486832981 | 0.05663816146 |
| 1 | 6  | 0.316227766   | 0.7608130253  |
| 2 | 10 | 1.58113883    | 0.05          |

## Zusammenfassung

Statt viele kleine, aufeinander abgestimmte  
edl-Skripten nacheinander auszuführen,  
kann man dies in einem Skript  
zusammenfassen und auch die Abstimmung  
(richtige Anzahlen, richtige Dateinamen  
etc.) miterledigen.



# edl-SkriptHeader

```
PROGRAM LEVEL 1.3    { Diese erste Zeile ist obligatorisch }
(
  Titel      = "Beispielfunktion",
  Keywords   = "Beispiel, MLP, Backprop",
  Rem1       = "Beispiel-EDL-Skript für NN-Übung",
  Rem2       = "Wendeparabel mit MLP lernen",
  Rem3       = "Hier: [-1,+1] auf Wendeparabel abbilden",
  Rem4       = "Datenvorverarbeitung"
);
{ Autor & Datum }
{ Zum Betrachten: Tabulatorbreite des Texteditors auf 8 Zeichen
  einstellen! }
{Kommentare stehen in geschweiften Klammern, sie dürfen i. d. R.
  überall dort stehen,wo auch Leerzeichen erscheinen können}
```

# edl-Deklarationen

```
{ In EDL (Experiment-Description-Language gibt es verschiedene EDL-
  Variablentypen, etwa STRING, LIST OF STRING, NUMBER und LIST OF NUMBER;
  grundsätzlich könnte man auch Skripten ohne solche Variablendeklarationen
  bzw. -definitionen schreiben, jedoch empfiehlt es sich, hier im Kopfteil
  des Skriptes (vor dem Schlüsselwort BEGIN) Variablen Werte zuzuweisen, die
  sich dann einfach überblicken und verändern lassen.}

STRING      Experiment      = "wende";
STRING      SubName         = "-1"; { Nummer des Subexperiments }

{ Die Ein- und Ausgabekanäle (-variablen) für das Backprop-(BP)-Netz (hier
  nicht vertippen!); die Namen müssen exakt mit denen in der RDF-Datei
  übereinstimmen }

LIST OF STRING InputChannels = [ "x", "Farbe" ];
LIST OF STRING TargetChannels = [ "f(x)" ];

{ Lektions/Datei-Namen }
STRING BasicName             = Experiment;
STRING Quell_RDF_Datei      = "Daten.rdf"; { mit txt2rdf.exe erzeugt }
STRING ScaledOutput          = "scaled.idf"; { Skalierte Daten,
  Incremental-Data-File }
STRING SelChannelsFile       = "selected.idf";
```

# edl-Dateinamen

```
{ Lektions/Datei-Namen }
STRING BasicName           = Experiment;
STRING Quell_RDF_Datei     = "Daten.rdf"; { mit txt2rdf.exe erzeugt }
STRING ScaledOutput        = "scaled.idf"; { Skalierte Daten,
    Incremental-Data-File }
STRING SelChannelsFile     = "selected.idf";
STRING LessonSet           = "lesson.idf"; { Trainingsdaten }
STRING TestSet             = "test.idf";   { Testdaten }
STRING LearnPAT            = BasicName + "l.pat"; { Trainingsdaten im
    Pattern-Dateiformat }
STRING TestPAT             = BasicName + "t.pat"; { Testdaten im
    Pattern-Dateiformat }
STRING LessonSource        = "lesson.idf";
}
```

# edl-Hilfsvariablen

```
{ Hilfsvariablen }
NUMBER START_IDX           = 0;
NUMBER LESSONSAMPLE_STEP   = 3; { 3 Samples lernen }
NUMBER TESTSAMPLE_STEP     = 1; { 1 Sample testen }
STRING Kommentar;
NUMBER i=1;
NUMBER Samplenr=1;
NUMBER LIST_END=1;
NUMBER NumLessonSamples=0;
NUMBER NumTestSamples=0;
BOOLEAN IsLessonSample = TRUE;
NUMBER Switchnr;
NUMBER NumOfSamples;
LIST OF NUMBER LessonSampleSet = [ 1 ];
    { wird unten erzeugt, muß hier aber initialisiert werden }
LIST OF NUMBER TestSampleSet   = [ 2 ]; { dto. }
NUMBER Epsilon                = 0.001;
    { Zur Vermeidung von Domain-Fehlern beim Skalieren }
```

# edl-Ausgabeveriablen

```
{ Die Programme, die in diesem Skript aufgerufen werden, sog.
  EDL-Klienten, liefern Ergebnisse zurück, die mit den
  folgenden Konstruktionen abgefragt werden können}

SECTION StdResult (
  BOOLEAN   Error;
  STRING    ErrorMessage;
); { SECTION StdResult }

SECTION ScaledResult (
  BOOLEAN   Error;
  STRING    ErrorMessage;
  NUMBER    NumOfSamples;
); { SECTION ScaledResult }

{ ----- }
```

# edl-SkriptHauptteil

```
BEGIN
{
  Überblick über die Vorverarbeitungsschritte hier:
  Auswahl der Kanäle x, Farbe und f(x) aus der RDF-Datei Daten.rdf
  (Select)
  Skalierung der Ein- und Ausgabekanäle (Scale)
  Erstellen von je einer Liste von Indizes von Trainings- und
  Testsamples
  Aufteilung der Samples in Trainings- und Testdaten nach diesen
  Listen (Select)
  Vorbereitung der Lektionen für das Programm "Backprop"(MkLesson)}
{ Wie man die Aufrufe der einzelnen EDL-Klienten (Programme) im
  EDL-Skript formatiert, spielt keine große Rolle; wichtig ist, daß
  die Darstellung konsequent und übersichtlich ist (Beispiele bei
  den nächsten beiden Aufrufen von "Select" und "Scale"!}
{ Hinweis: die Bedeutung der Optionen der einzelnen Klienten ist in
  der jeweiligen Online-Hilfe dokumentiert; in vielen Fällen
  erklären sie sich aber auch von selbst! }
```

# edl-Select-Aufruf

```
{ Auswählen der Kanäle; hier x, Farbe und f(x) (vgl. RDF-Datei) }
  CALL "Select" ( Input      = Quell_RDF_Datei,
                  Output     = SelChannelsFile,
                  Channels   = InputChannels + TargetChannels,
                  Epsilon
                )
  GIVING StdResult;
```

# edl-Scale-AufrufInput

```
{ Skalierung (z-Transformation) der Eingabedaten }
  CALL "Scale" (
    Input      = SelChannelsFile,
    Output     = ScaledOutput,
    Channels   = InputChannels,
    Method     = "Linear",
    Base       = "Internal",
    Domain     = "Mean",
    Range      = [-1,1],
    KeepInput  = TRUE,
    Epsilon
  )
  GIVING ScaledResult;
```

# edl-Scale-AufrufOutput

```
{ Abbildung der Ausgabedaten auf das Intervall [0.05,0.95] }
    CALL "Scale" (
        Input      = ScaledOutput,
        Output     = "temp.idf",
        Channels    = TargetChannels,
        Method     = "Linear",
        Base       = "Internal",
        Domain     = "Max",
        Range      = [0.05,0.95],
        KeepInput  = TRUE,
        Epsilon
    )
    GIVING ScaledResult;
{ Die Anzahl der Daten (Samples) in der RDF-Datei als Rückgabewert
  des EDL-KLienten "Scale": }
    NumOfSamples = ScaledResult.NumOfSamples;
```

# edl-AufteilungderIndices

```
{ Aufteilung der Listen von Indizes der Samples in Trainings- und Testdaten: }
{ Dieser Teil ist nicht so wichtig und muß nicht beim ersten Lesen verstanden werden }
    Samplenr=START_IDX;
    Switchnr = Samplenr + LESSONSAMPLE_STEP - 1;          {ist schon richtig so ...}
    IsLessonSample = TRUE; WRITE(Switchnr);
    WHILE (Samplenr<=NumOfSamples)
    DO BEGIN
        IF (IsLessonSample==TRUE)
        THEN BEGIN NumLessonSamples=NumLessonSamples+1;          { Resize Liste: }
            LessonSampleSet = LessonSampleSet[1,NumLessonSamples];
            LessonSampleSet[NumLessonSamples,1] = [Samplenr]; END
        ELSE BEGIN
            NumTestSamples=NumTestSamples+1;
            TestSampleSet = TestSampleSet[1, NumTestSamples];
            TestSampleSet[NumTestSamples,1] = [Samplenr]; END;
        IF (Samplenr>=Switchnr)
        THEN BEGIN IF (IsLessonSample)
            THEN BEGIN Switchnr=Switchnr+TESTSAMPLE_STEP;IsLessonSample=FALSE;END
            ELSE BEGIN Switchnr=Switchnr+LESSONSAMPLE_STEP; IsLessonSample=TRUE; END
            END;
        Samplenr=Samplenr + 1;
    END;
```

# edl-Aufteilung der Samples

```
{ Aufteilung der Daten in Lektion und Testset }
{ Trainingssamples }
    CALL "Select" ( Input          = "temp.idf",
                    Output         = LessonSet,
                    Channels       = ["Winkel","Sinus"],
                    SampleSet      = LessonSampleSet,
                    KeepInput      = TRUE )
    GIVING StdResult;
{ Testsamples }
    CALL "Select" ( Input          = "temp.idf",
                    Output         = TestSet,
                    Channels       = ["Winkel","Sinus"],
                    SampleSet      = TestSampleSet,
                    KeepInput      = TRUE )
    GIVING StdResult;
```

# edl-Lektionsaufbau

```
{ Aufbau der Lektion }
    CALL "Mklesson"
    ( Input          = LessonSet,
      Target         = TargetChannels,
      KeepInput      = TRUE,
      Output         = LearnPAT )
    GIVING StdResult;
    CALL "Mklesson"
    ( Input          = TestSet,
      Target         = TargetChannels,
      KeepInput      = TRUE,
      Output         = TestPAT )
    GIVING StdResult;
{ man kann in das Protokoll-File auch andere Dinge schreiben, z.B: }
    WRITE ( Ende = "Fertig" );
END. { PROGRAM }
```

# Dateiumwandlung

- Ascii-datei .dat,.txt  
– txt2rdf
- RohDatenFormat .rdf  
– incrementelleFilter
- IncDatenFormat .idf  
– Mklesson
- Pattern .pat

## 1. bp.edl

```
PROGRAM LEVEL 1.3
( Titel      = "...",
  Rem4       = "Backprop aufrufen");
STRING      Experiment      = "wende";
STRING      SubName        = "-bp";      { Bezeichnung des Subexperiments
}
{ Steuerung Netzwerk (BACKPROP) }
NUMBER      MaxCycles       = 2000; { max. Anzahl der Lerndurchläufe }
LIST OF NUMBER Topology = [ 1, 8, 1 ];
      {1 Eingabeneuron , 8 versteckte Zellen und ein Ausgabeelement }
{ SteuerParameter }
NUMBER      Lambda          = 2.1,      { vernünftige Wahl }
            Eta              = 0.9,      { initiale Schrittweite }
            Alpha            = 0.9,      { initialer Momentum-Wert }
            Psi               = 1.02,    { kaum Verschlechterung zugelassen }
            MinError          = 0.001,   { Abbruchbedingung }
            BackupSteps       = 500,
            OutputSteps       = 30,
            MonteCarlo        = 10;

LIST OF NUMBER LMDTolerance   = [ 50, 0.00001 ]; { Abbruch des Trainings, wenn
      sich der Fehler in 50 Schritten um weniger als 0.00001 verringert }
LIST OF NUMBER RandomRange    = [ -3, 3 ];      { Initialer Bereich der Gewichte }
```

## 2. bp.edl

```
{ Lektions/Datei-Namen }
{ ----- }

STRING BasicName           = Experiment;
STRING LessonName          = BasicName + "l.pat"; { wie in wene.edl }
STRING TestSetName         = BasicName + "t.pat";
STRING NetFileName         = BasicName + SubName + ".net";
STRING LernProtokoll       = BasicName + SubName + ".lp";
STRING LektionsTestProtokoll = BasicName + SubName + ".wp";
STRING TestSetTestProtokoll = BasicName + SubName + ".tp";
SECTION StdResult ( BOOLEAN Error; STRING ErrorMessage; ) { SECTION StdResult }
{ Antwort des Netzes }
SECTION BackpropResult (
    STRING Result;          { "LEARNED", "TESTED", "WORKED", "NOT_LEARNED", ... }
    BOOLEAN Error;
    NUMBER NetError;        { Zur Zeit der halbe Mean-Squared-Error }
    NUMBER Cycles;          { Anzahl von Trainingsschritten }
    NUMBER RMSError;        { Zur Zeit der halbe Mean-Squared-Error }

); { SECTION BackpropResult }
```

## 3. bp.edl

```
BEGIN { Erzeuge leeres Netz }
CALL "MkNet"(
    NetFile      = NetFileName,
    NetworkType  = "Backprop",
    ActFunc      = "Fermi",
    Topology )
GIVING StdResult;
{ Trainiere Netz }
CALL "backprop" (
    NetFile      = NetFileName,
    PatFile      = LessonName,
    OutFile      = LernProtokoll,
    BackupSteps,
    MaxCycles,
    OutputSteps,
    MinError,
    MonteCarlo,
    Alpha, Lambda, Eta, Psi,
    RandomRange, LMDTolerance,
    Quiet        = FALSE )
GIVING BackpropResult;
WRITE( AnzSchritte = BackpropResult.Cycles );
```



## 4. bp.edl

```
CALL "backprop" (                                { Test auf Lektion }
  NetFile = NetFileName,
  PatFile = LessonName,
  OutFile = LektionsTestProtkoll,
  Quiet   = TRUE,
  Mode    = "WORK" )
GIVING BackpropResult;
WRITE( Trainingsfehler = BackpropResult.NetError );
WRITE( RMSTrainingsfehler = BackpropResult.RMSError );
WRITE( Ende = "Training abgeschlossen", Experiment );
CALL "backprop" (                                { Test auf Test-Set }
  NetFile = NetFileName,
  PatFile = TestSetName,
  OutFile = TestSetTestProtokoll,
  Quiet   = TRUE,
  Mode    = "WORK" )
GIVING BackpropResult;
WRITE( Testfehler      = BackpropResult.NetError );
WRITE( RMSTestfehler   = BackpropResult.RMSError );
WRITE( Ende = "Test abgeschlossen",Experiment, SubName);  END.
```

## bp.prt

|                       |                           |               |                    |       |
|-----------------------|---------------------------|---------------|--------------------|-------|
| [BackProp]            |                           | MaxCycles     | 2500               |       |
| Version:              | BackProp 5.4              | MinError      | 0.001              |       |
| Start-Time:           | 18.05.2000 12:41:54       | LMDTolerance  | 100                | 0.001 |
| [BackProp-Parameters] |                           | MonteCarlo    | 10                 | 0     |
| NetType               | CEnhancedBackNet          | OutputSteps   | 10                 |       |
| Mode                  | WORK                      | BackupSteps   | 100                |       |
| Terminator            | CNetTerminator            | Quiet         | true               |       |
| Metric                | C1DimQuadraticErrorMetric | ActFunc       | Fermi              |       |
| LSControl             | true                      | [EDL-WRITE]   |                    |       |
| GlobalStepMode        | true                      | Testfehler    | 0.02218824977      |       |
| NetType               | CEnhancedBackNet          | RMSTestfehler | 0.2106573036       |       |
| NetFile               | wende-bp.net              | Ende          | Test abgeschlossen |       |
| OutputNetFile         | wende-bp.net              | Experiment    | wende              |       |
| PatFile               | wendet.pat                | SubName       | -bp                |       |
| OutFile               | wende-bp.tp               |               |                    |       |

# wende-bp.lp

```
[MonteCarlo.Channels]
  MCStepError

[MonteCarlo.Protocol]
  0      0.04307707736

[MonteCarlo.Winner]
  BestError      0.04307707736

[Learning.Channels]
  Iterations      TotalError      LSC-evaluation      eta

[Learning.Protocol]
  0      0.04307707736      0.9
  30     0.00676630861      Improved  0.3881681992
  60     0.003263958984     Improved  0.2790269461
  90     0.001907396279     Improved  0.2406875271
  120    0.001296823778     Accepted  0.6920536591
  136    0.0009736168648    Improved  2.221351179
```

# wende-bp.tp

```
[Description]
  Laboratory      FGNN
  OriginalFilename BackProp
  Translator      <<UNKNOWN>>
  TranslationDate 18.05.2000 12:41:54
  NumberOfChannels 3
  DataType      Incremental Data File (IDF)
  DeltaT1
  SamplingDate   18.05.2000 12:41:54

[Channels]
  Time  Output.0      Target.0      Error

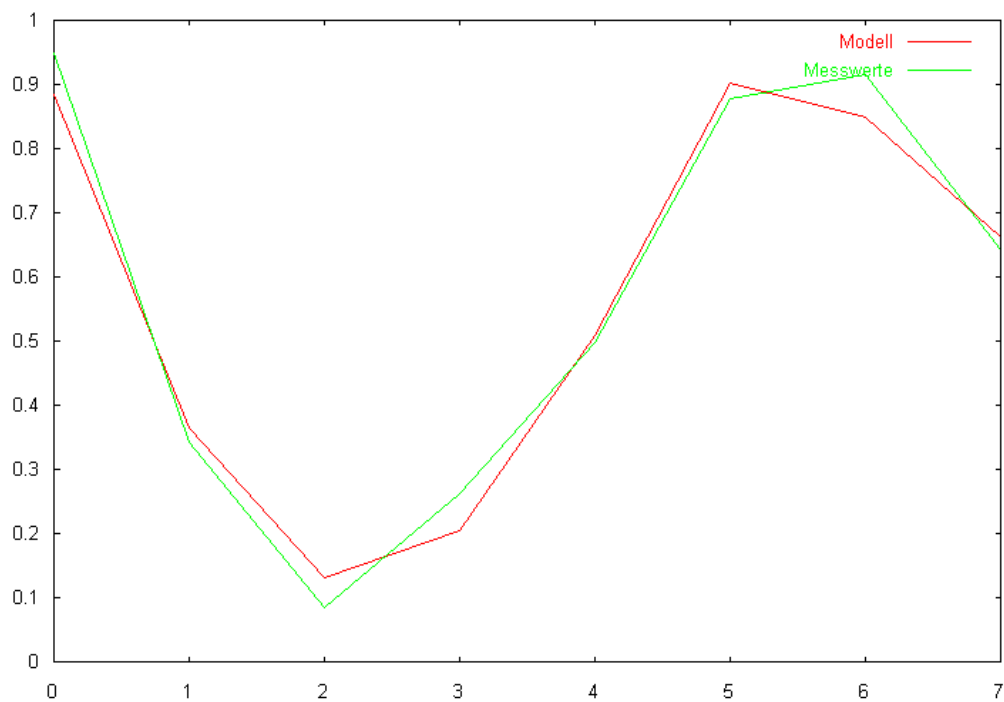
[Data]
  0      0.1426675747      0.05663816146      0.003700529972
  1      0.8398357373      0.7608130253      0.003122294506
  2      0.3956643599      0.05      0.05974192484

[Statistics]...
```

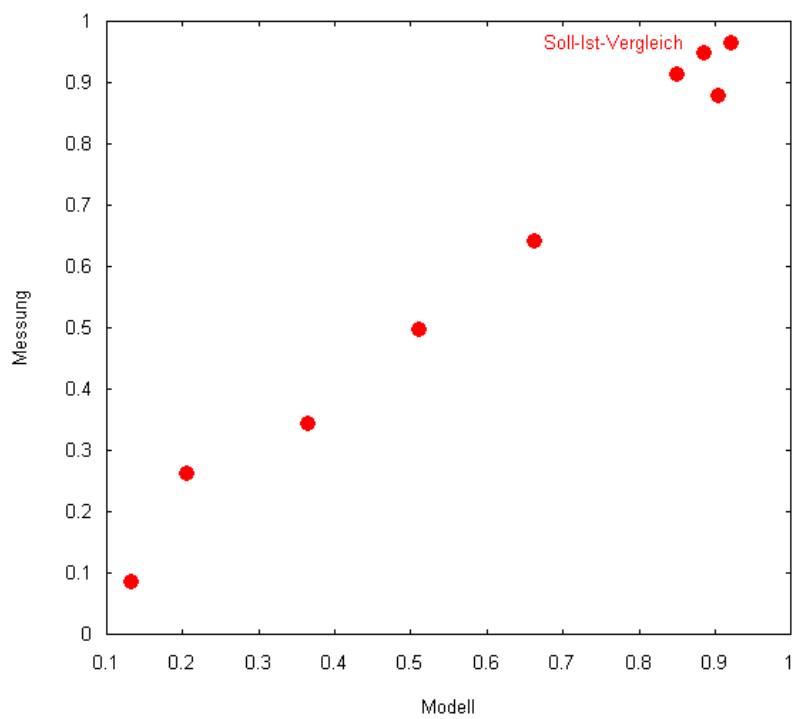
# wende-bp.wp

```
[Description]
  Laboratory      FGNN
  OriginalFilename BackProp
  Translator      <<UNKNOWN>>
  TranslationDate 18.05.2000 12:41:52
  NumberOfChannels 3
  DataType        Incremental Data File (IDF)
  DeltaT1
  SamplingDate    18.05.2000 12:41:52
[Channels]
  Time  Output.0      Target.0      Error
[Data]
  0      0.8840450958      0.95      0.002175024696
  1      0.3639064127      0.343616204      0.0002058462838
  2      0.1322237883      0.08588059803      0.001073845644
  3      0.2059278604      0.2632596455      0.001643466793
  4      0.5097396292      0.4977505507      7.186900124e-05
  5      0.9033545368      0.8784449236      0.000310244415
  6      0.8498262323      0.9150028743      0.00212399733
  7      0.6621168589      0.6429001603      0.000184640753
[Statistics] ...
```

## Bildausgabe1



# Bildausgabe2



# wende-bp.net

|                               |              |                |               |              |
|-------------------------------|--------------|----------------|---------------|--------------|
| [CEnhancedBackNet.Topology]   |              |                |               |              |
| Layers 3                      |              |                |               |              |
| Thresholds true               |              |                |               |              |
| Neurons                       | 1            | 8              | 1             |              |
| ActFunc Fermi                 |              |                |               |              |
| [CEnhancedBackNet.Activities] |              |                |               |              |
| 0                             | 1.264911064  | -1             |               |              |
| 1                             | 0.5163736604 | 0.1946317398   | 0.0060321433  | 0.4596654616 |
|                               | 0.9654226915 | 5.14528458e-08 | 0.4544312831  | 0.9946157491 |
|                               | -1           |                |               |              |
| 2                             | 0.6621168589 |                |               |              |
| [CEnhancedBackNet.Weights]    |              |                |               |              |
| 0,0                           | 1.958087415  | 1.218972496    | -4.279413887  | -2.662002689 |
|                               | 3.736459149  | -6.245700971   | -2.726540149  | 4.373550231  |
| 0,1                           | 2.411288367  | 2.962082177    | -0.3084754912 | -3.205507158 |
|                               | 1.396920134  | 8.882343752    | -3.266048744  | 0.3132737872 |

-> wende-bp.net

```
1,0    -3.109272512
1,1     0.4588250607
1,2    -3.078534618
1,3     1.3905256
1,4     0.7677624709
1,5     5.66921999
1,6     2.273727572
1,7     2.492221083
1,8     2.684890837
[CEnhancedBackNet.Parameter]
RandomRange    -3      3
Alpha          0.9
Eta            2.221351179
WeightDecay    0
Beta           0.5
Psi            1.02
Phi            1.2
Lambda         2.1
MinEta         0.001
BestError      0.0009736168648
```

## Netze

- Eingabe von Patterns in Netzwerke erzeugt verschiedenen Protokolldateien
  - .LP      Lern-Protokoll
  - .WP      Work-Protokoll
  - .TP      Test-Protokoll
- Die Protokolle können ausgegeben werden
  - mit wp2txt      als Ascii-datei
  - mit plotwp      als Bildschirmplot  
als EPS-datei