Neuronale Netze, Fuzzy Control, Genetische Algorithmen
Prof. Jürgen Sauer
Lehrbrief Nr. 12: Neuro-Fuzzy Systeme

1. Einführung in Neuro-Fuzzy-Systeme

1.1 Übersicht

Komplementäre Eigenschaften der beiden in einem Neuro-Fuzzy-System zusammengefassten Methoden spielen eine Rolle:

- Neuronale Netze beziehen ihr Wissen in erster Linie aus Daten.
- Fuzzy-Systeme setzen einen Experten voraus

Neuronale Netze bilden durch Trainingsmuster einen ihnen angemessene Struktur heraus, die es anschließend erlaubt, bis dahin unbekannte Testmuster zu verarbeiten. Die Testmuster, die ähnlich strukturiert sind wie die Trainingsmuster, führen nach der Verarbeitung durch das Neuronale Netz zu ähnlichen Resultaten wie die Trainingsmuster. Mit der Fuzzy Logik werden nahe beieinander liegende Punkte des Eingaberaums ähnlich – d.h. in nahe beiander liegende Punkte des Ausgaberaums abgebildet und verarbeitet. Dadurch vermeidet diese Logik Sprünge und scharfe Kanten, die nebeneinander liegende Punkte eindeutig voneinander trennen würden. Auch hier können wieder nahe beinander liegende Punkte als Daten mit ähnlicher Struktur gedeutet werden, die durch überlappende Zugehörigkeitsfunktionen und Regeln sowie durch die interpolierenden Eigenschaften des Defuzzifizierers ähnlich verarbeitet werden.
Mit dem Begriff Neuro-Fuzzy werden solche Systeme belegt, die ein Fuzzy-System und ein Neuronales Netz getrennt voneinander zur Bewältigung von Teilen desselben Problems einsetzen.
Es ist naheliegend, Neuronale Netze zur Bestimmung der Parameter eines Fuzzy-Systems heranzuziehen und somit durch Kombination dieser beiden Ansätze ein lernfähiges Neuro-Fuzzy System zu schaffen. Auf diese Weise erhält man ein transparentes System, dessen Struktur durch Fuzzy-Regeln geprägt ist und adaptiv ist, d.h. seine Parameter selbstständig an seine Aufgabe anpassen kann. Bei den Parametern handelt es sich um Anzahl und Art der Fuzzy-Regeln sowie um Fuzzy-Mengen, die gewöhnlich durch parametrisierte Dreiecks-, Trapez- oder Gaußfunktionen modelliert werden.
Es gibt zahlreiche Ansätze zu Neuro-Fuzzy-Systemen. Es lassen sich kooperative und hybride Ansätze unterscheiden. Beiden **kooperativen** Ansätzen werden Neuronale Netze zur Bestimmung eines Teils der Fuzzy-Parameter herangezogen. Nach deren Bestimmung wird ein herkömmliches Fuzzy-System erstellt, das ohne das Neuronale Netz arbeitet. **Hybride** Ansätze verschmelzen das Fuzzy-System mit dem Neuronalen Netz. Auf diese Weise entsteht eine Architektur, die sowohl als spezielles Neuronales Netz als auch als gewöhnliches Fuzzy-System arbeiten kann. Eine Trennung der beiden Einzelsysteme ist nicht mehr möglich.

1.2 Verschiedene Neuro-Fuzzy-Ansätze

ANFIS[1] (Adaptive Network-based Fuzzy-Inferenz-System) implementiert Fuzzy Regeln in einem Neuronalen Netz, das aus 6 Schichten besteht, die allerdings nicht homogen aufgebaut sind. Die Neuronen verschiedener Schichten verwenden jeweils verschiedene Aktivierungsfunktionen (z. B. Gaußfunktionen, usw.). Die Zugehörigkeitsfunktionen von Fuzzy-Prädikaten werden bspw. durch Gaußfunktionen realisiert. Das Lernen des Reglers geschieht nur in wenigen Schichten. Die richtigen Zugehörigkeitsfunktionen müssen von Anfang an bekannt sein, während die Parameter der Konklusionen (Sugeno-Form) gelernt werden.
BIOFAM (Basic Input Output Fuzzy Associative Map)[2] ist ein System, das Kosko in seinem Buch[3] vorschlägt. BIOFAMs führen eine Abbildung von einem einzelnen Eingabewert auf eine Fuzzy-Verteilung durch. Für jeden Term der Regelprämisse wird ein eigenes FAM (Fuzzy Adaptive Map)[4] gebildet, das eine Regel bis auf die Verknüpfung der Prämissenterme repräsentiert. Die FAMs können dabei auch durch zwei verschiedene Algorithmen gebildet werden. Einer davon verwendet ein Neuronales Netz. Das Neuronale Netz dient nur zur Entwicklung des Systems, hinterher wird das gelernte Wissen in einem Fuzzy-System implementiert. Kosko muß seine automatisch gefundenen Regeln teilweise mit Regeln aus wenig repräsentativen Datenbereichen ergänzen, da die Häufigkeitsverteilung der Daten das Gewicht einer Regel festlegt. Dies ist insbesondere bei regeltechnischen Aufgaben unerwünscht.
NEFCON (Neural-Fuzzy Controller)[5] ist ein dreischichtiges Neuronales Netz mit Spezialneuronen (Fuzzy-Perzeptron), das einen Fuzzy Controller repräsentieren kann. Gewichte sind allerdings keine Zahlenwerte, sondern Fuzzy-Mengen. Für die Anpassung der Gewichte wird Backpropagation oder ein anderes gebräuchliches Gradientenverfahren benutzt. Für die Beurteilung einer Aktion muß eine Gütefunktion definiert und ihre Beurteilung auf die einzelnen Regeln zurückgeführt werden. Für jeden einzelnen Lernschritt muß angegeben werden, ob das Vorzeichen der resultierenden Aktion mit dem Vorzeichen der optimalen Aktion übereinstimmt. Diese Vorschrift ist leichter zu erfüllen als eine genaue Angabe der optimalen Aktion. Trotzdem ist in vielen nichtlinearen Systemen höherer Ordnung das Vorzeichen der optimalen Aktion nicht so leicht zu ermitteln.
Ein Fuzzy Perzeptron ist von einem dreischichtigen Perzepton abgeleitet. Dabei werden die Gewichte auf den Verbindungen durch Fuzzy-Mengen ersetzt (Fuzzy-Gewichte) und die Aktivierungs- und Propagierungsfunktion so modifiziert, dass sie mit Fuzzy Mengen operieren können. Die Eingabeinheiten repräsentieren die Eingangsgrößen, die inneren Einheiten stehen für die Fuzzy-Regeln, und die Ausgabeeinheiten stellen die Stellgrößen dar.

[1] Vgl. Skriptum 3.2.3.4.2
[2] Vgl. Skriptum 3.1.5.4
[3] Kosko, Bart: Neural Networks and Fuzzy Systems, Prentice Hall, Englewood Cliffs, New Jersey, 1992
[4] vgl. Skriptum, 3.1.5
[5] Das NEFCON-Modell wurde Mitte der 90er Jahre an der T.U. Braunschweig von D. Nauk, F. Klawonn und R. Kruse entwickelt.

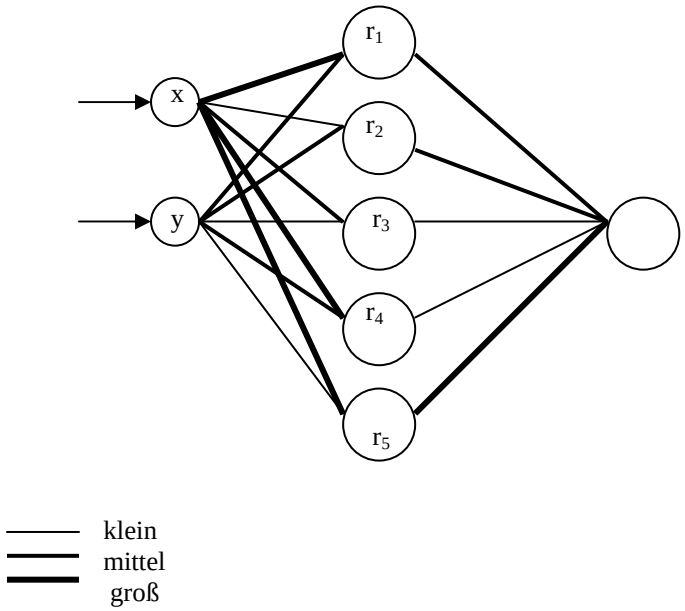


Abb.: Hybrides Neuro-Fuzzy-System nach dem NEFCON-Modell. Die Verbindungslinien zwischen den Neuronen entsprechen unscharfen Mengen. Durch die Strichstärke der Verbindung ist die Zugehörigkeit zu einer besimmten Fuzzy-Menge ausgedrückt. Die Neuronen der Eingabeschicht entsprechen den Eingangsgrößen, die der inneren Schicht den Fuzzy-Regeln, die der ausgangsschicht der Ausgangsgröße. Die Regel2 lautet somit: „Wenn x klein ist und y groß, dann ist z mittel“.

Die Propagierungsfunktion zwischen Eingabeschicht und innerer Schicht bestimmen die Zugehörigkeitsgrade der Eingaben zu den Fuzzy-Gewichten der entsprechenden Verbindungen. Durch eine Minimumbildung bestimmt die Aktivierungsfunktion einer inneren Einheit daraus den Erfüllungsgrad einer Regel. Die Propagierungsfunktionen verknüpfen daraufhin die Erfüllungsgrade mit den Fuzzy-Gewichten der zur Ausgabe führenden Verbindungen. Die Aktivierungsfunktion einer Ausgabe-Einheit fasst die modifizierten Fuzzy-Mengen zusammen und führt anschließend die Defuzzifizierung durch.

2. ANFIS

2.1 Das Netz[6]

ANFIS ist ein Netz mit 6 feedforward Schichten, wobei keine Verbindung besonders gewichtet wird, d.h. die Logik steckt einzig und alleine in den Neuronen und der Verknüpfungstopologie. Das Neuronale Netz kann die Funktionalität eines Sugeno-Reglers[7] bereitstellen.
Für die erste Schicht wird meistens eine Glockenfunktion mit Maximum 1 und Minimum 0 für die Zugehörigkeitsfunktion μ benutzt. Die Regelbasis eines Sugeno-Typs kann bspw. so formuliert werden:

IF x is A_1 and y is B_1 THEN $f_1 = p_1 x + q_1 y + r_1$
IF x is A_2 and y is B2 THEN $f_2 = p_2 x + q_2 y + r_2$

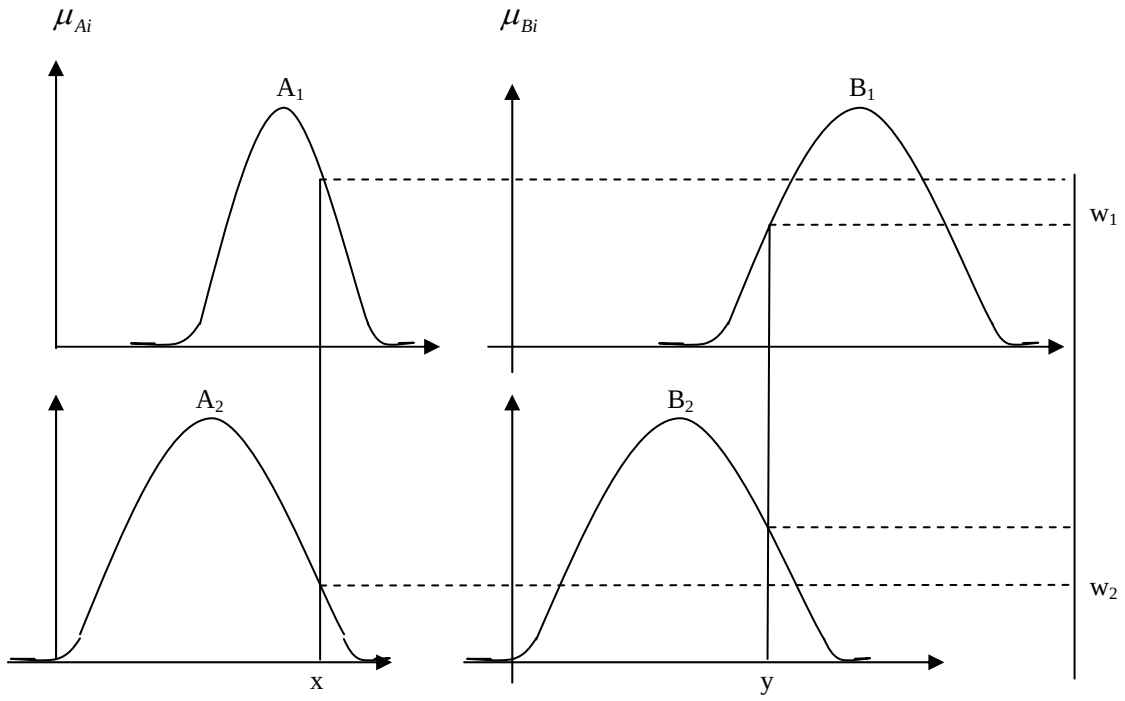


Abb.: Anfis Netz für Typ 3 Fuzzy Contoller

Die Anwendung der Regeln erfolgt mit dem logischen AND (“product for T-norm”):

1. Auswertung der Regelprämissen $w_i = \mu_{A_i}(x)\mu_{B_i}(y) \quad i = 1,2$
2. Auswertung der Implikation und der Regelkonsequenzen:
   $$f(x,y) = \frac{w_1(x,y)\cdot f_1(x,y) + w_2(x,y)\cdot f_2(x,y)}{w_1(x,y) + w_2(x,y)}$$ bzw. ohne Argumente
   $$f = \frac{w_1 f_1 + w_2 f_2}{w_1 + w_2}$$

[6] Vgl. Skriptum 3.2.3.4.2.1
[7] vgl. Skriptum 3.1.4

Das kann aufgeteilt werden: $\bar{w}_i = \frac{w_i}{w_1 + w_2}$, $f = \bar{w}_1 f_1 + \bar{w}_2 f_2$

Alle Berechnungen können folgendermaßen zusammengefasst werden:

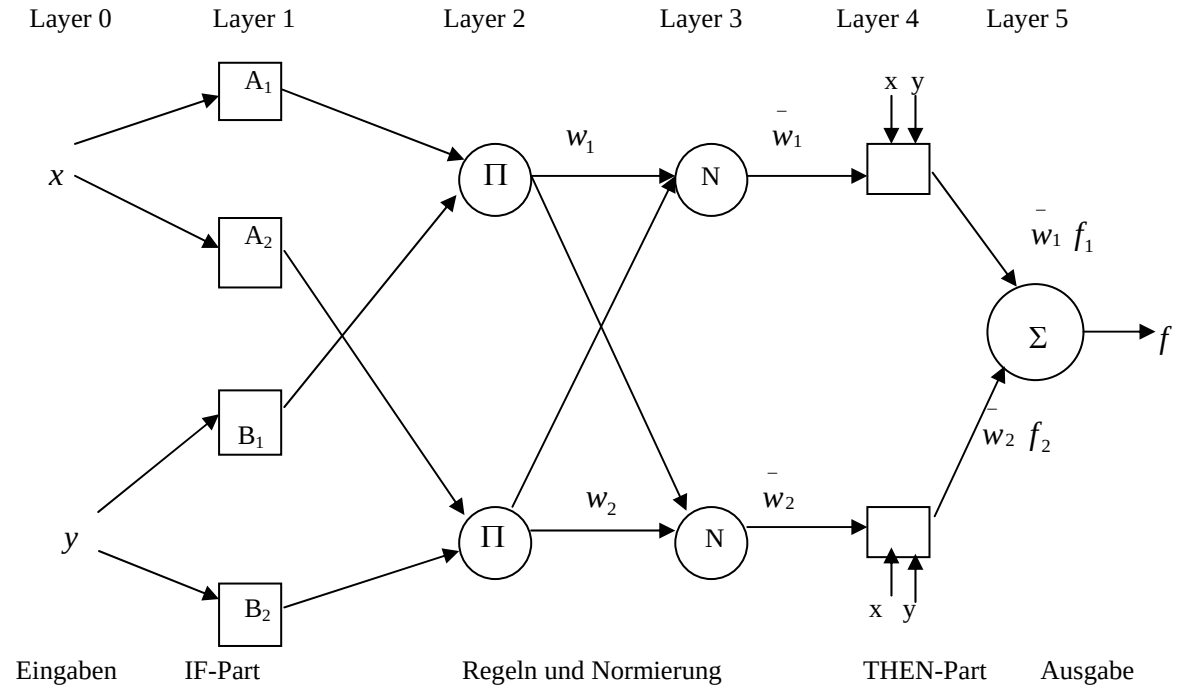


Abb.: Anfis-Netz für Typ 3 Fuzzy-Controller

In der 2. Schicht gibt es für jede Fuzzy-Regel genau ein Neuron zur Bildung der Regelprämisse, in dem einmündende Zugehörigkeitswerteaufmultipliziert werden (auch ein „Fuzzy AND“ bzw. Schnitt sind denkbar)
Die mittlere Neuronenschicht hat die gleiche Dimension wie ihre mit ihr maximal verbundene Vorgängerschicht. Sie kalkuliert den Erfüllungsgrad der Prämisse in Abhängigkeit aller anderen Regeln: $\bar{w}_i = \frac{w_i}{w_1 + w_2}$.

Die 4. Schicht (hier: viereckige Neuronen) hat zusätzlich zur Verbindung mit genau einem Neuron der 3. Schicht Verbindungen zu allen Eingabe-Neuronen. Hier wird die Konklusion der Regel berechnet, z.B.: $o_1 = \bar{w} \cdot (p_1 x + q_1 y + r_1)$. Die Parameter p_1, q_1 und r sind vom Lernverfahren zu bestimmen.

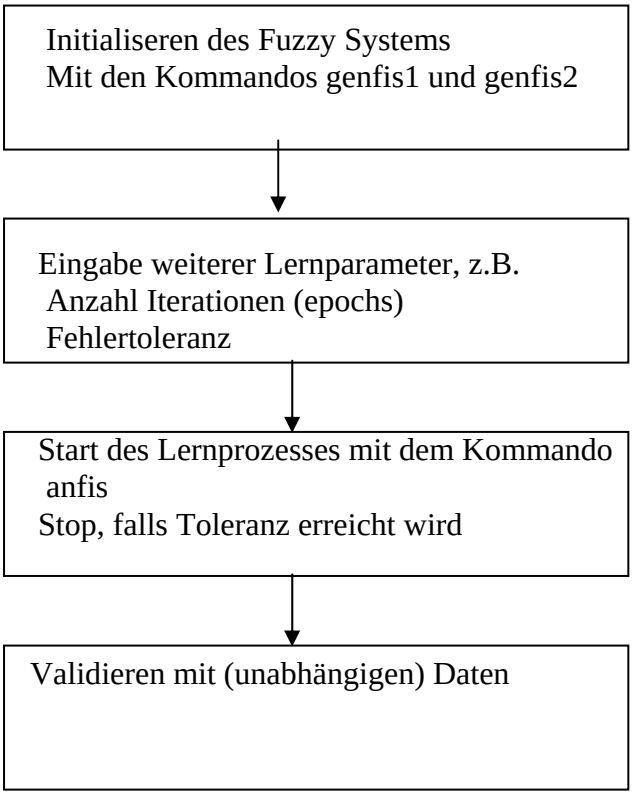
Lernregeln für ANFIS-Netze: Backpropagation und Hybrid-Regel
Für die Hybrid-Lernregel werden Parameter in 2 Arten aufgeteilt:

- Prämissenparameter S_1, die man durch das Backpropagationverfahren optimiert.
- Konsequenzparameter S_2, die man durch die Methoden des kleinsten quadratischen Schätzers bestimmt. Das funktioniert nach folgendem Ablauf:
- 1. Für jedes Trainingspaar ($\vec{x}$, output) berechnet man die Werte, so dass für das $\vec{x}$ gerade die Funktion des Netzes in Abhängigkeit der Werte von S_2 entsteht. Die Werte von S_1 sind vorgegeben und fest. Da die Funktion des Netzes linear von S_2 abhängt, ergibt sich für $s \in S_2$ und a ein 1xS_2-Vektor: $a \cdot s = output$. Das ist das Vorgehen für alle Trainingspaare und insgesamt erhält man Matrizen A, B mit $A \cdot s = B$

- 2. Da die Zahl der Trainingspaare meistens größer ist als die Mächtigkeit von S_2, ist das Problem überbestimmt, und es existiert möglicherweise keine eindeutige Lösung. Man behilft sich mit dem kleinsten quadratischen Schätzer, der $\|AS - B\|^2$ minimiert. Diesen berechnet man mit Pseudoinversen[8]. So kann man die Parameter von S_2 anpassen.
- 3. Die Parameter von S_1 werden mit Hilfe von Backpropagation bei festen Werten von S_2 optimiert.

2.2 Berechnungen in ANFIS mit MATLAB

In der Fuzzy Control Toolbox gibt es ein nützliches Kommando „`anfis`“. „anfis“ besitzt ein Optimierungsschema zur Bestimmung der Parameter von einem Fuzzy System, das an Daten sich bestens anpasst. Beinahe alle Algorithmen für Optimierungen nutzen die Berechnung der Gradienten, hier geschieht es mit einem Neuronalen Netz.



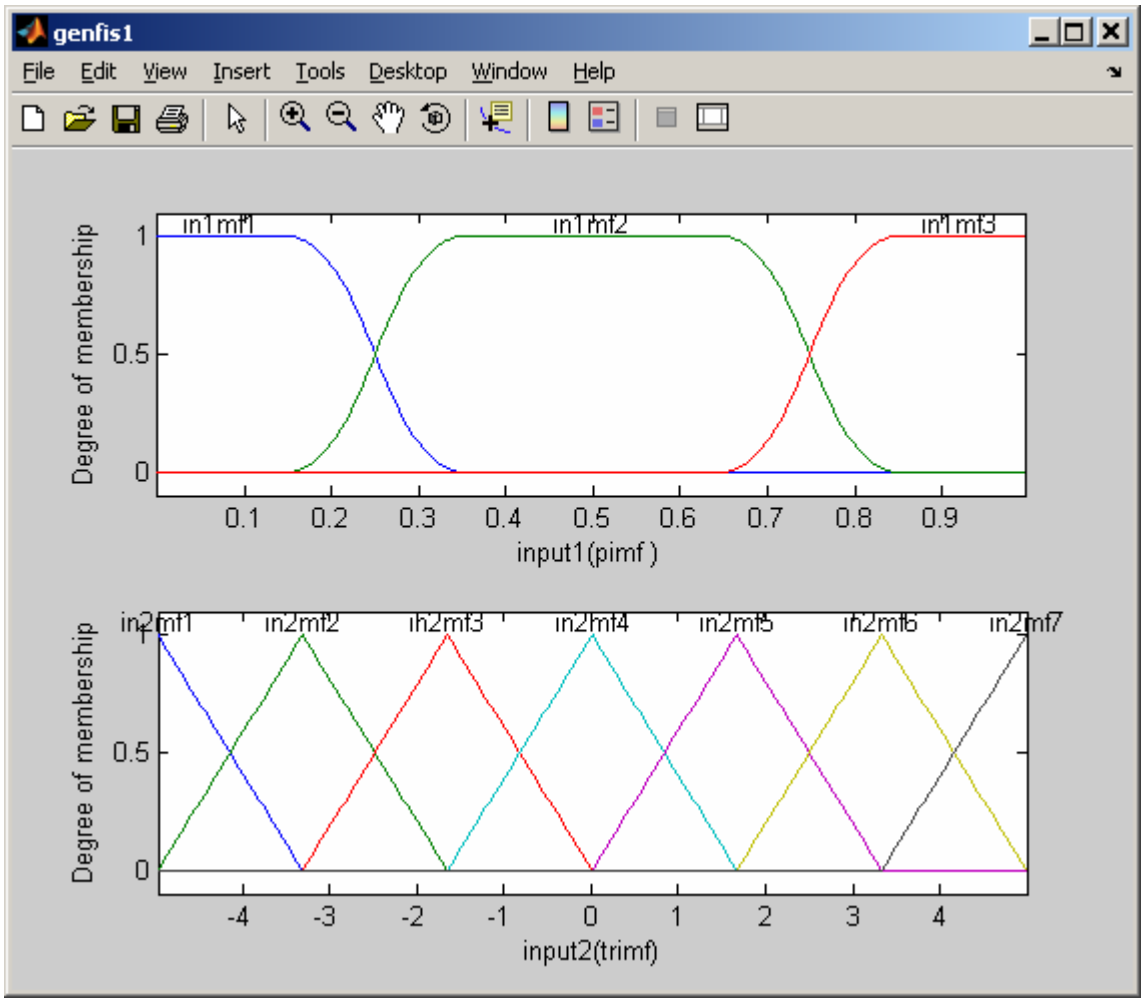
[8] Falls bspw. $A^T A$ invertierbar ist, dann ist das Pseudoinverse $\left(A^T A\right)^{-1} A^T$ und der kleinste quadratischje Schätzer $X^* = \left(A^T A\right)^{-1} A^T B$

GENFIS1 und ANFIS-Kommandos
`genfis1` erzeugt das initiale FIS-System, `anfis` erzeugt das optimale FIS-System.
Typ und Anzahl der Zugehörigkeitsfunktionen werden über das Kommando `genfis1` bestimmt:

GENFIS1(DATA,MF_N,IN_MF_TYPE.

Die FIS-Matrix wird mit Hilfe von genfis1 aus den Trainingsdaten DATA generiert. MF_N ist ein Vektor, der die Anzahl der Zugehörigkeitsfunktionen für die Eingabe spezifiziert. IN_MF_TYPE ist ein Zeichenkettenfeld, in der jede Zeile den MF-Typ einer Eingabe-Variable festlegt, z.B.:

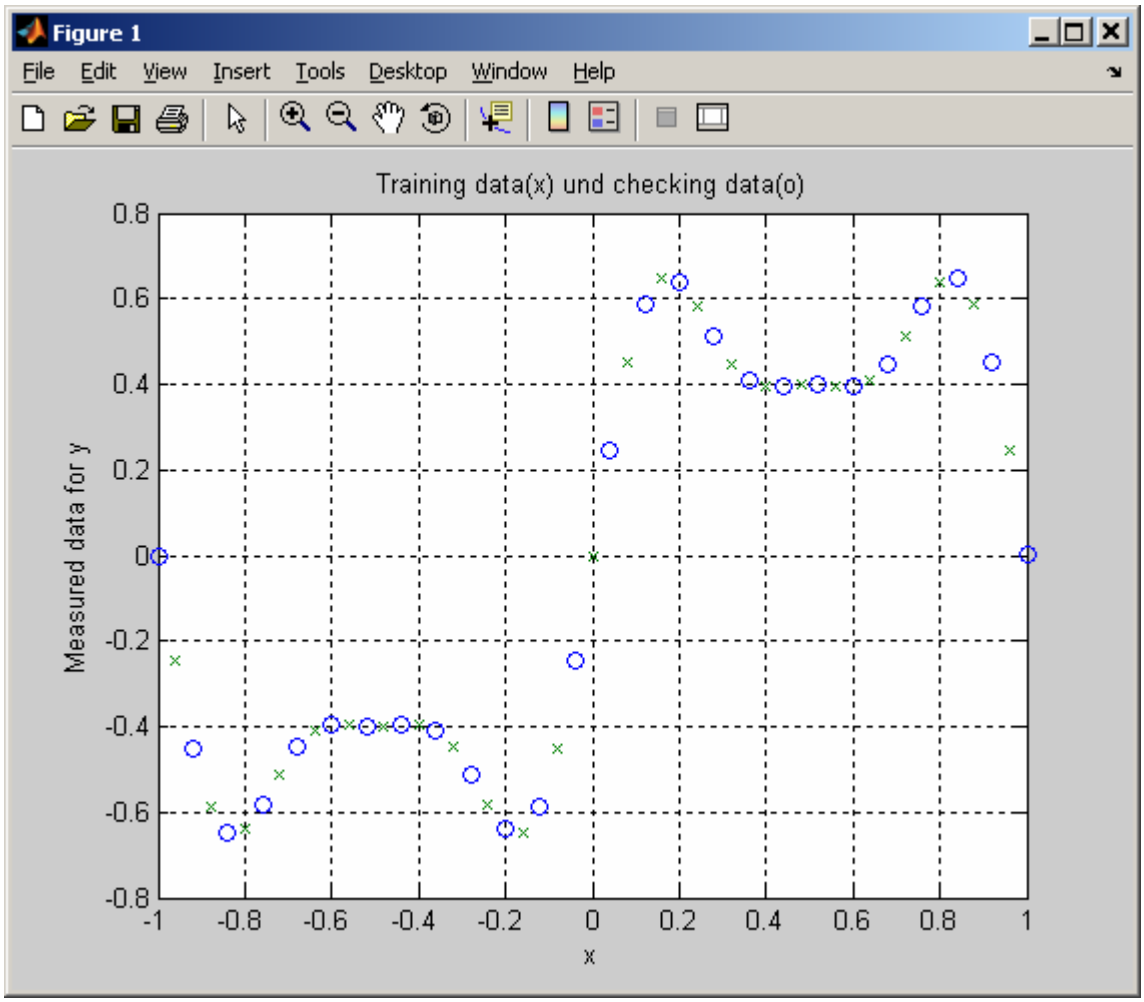
```
% Erzeugen Zufallszahlen
NumData = 1000;
data=[rand(NumData,1) 10*rand(NumData,1)-5 rand(NumData,1)];
% Spezifiziere Anzahl und Typ der Zugehoerigkeitsfunktion
NumMf=[3 7];
MfType=str2mat('pimf','trimf');
% Verwende genfis1 zur Erzeugung in initialer Zugehoerigkeitsfunktionen
FisMatrix=genfis1(data,NumMf,MfType);
% Zeichne die Zugehoerigkeitsfunktionen
figure('name','genfis1','numbertitle','off');
NumInput=size(data,2)-1;
for i=1:NumInput;
    subplot(NumInput,1,i);
    plotmf(FisMatrix,'input',i);
    xlabel(['input' num2str(i) '(' MfType(i,:) ')']);
end;
```



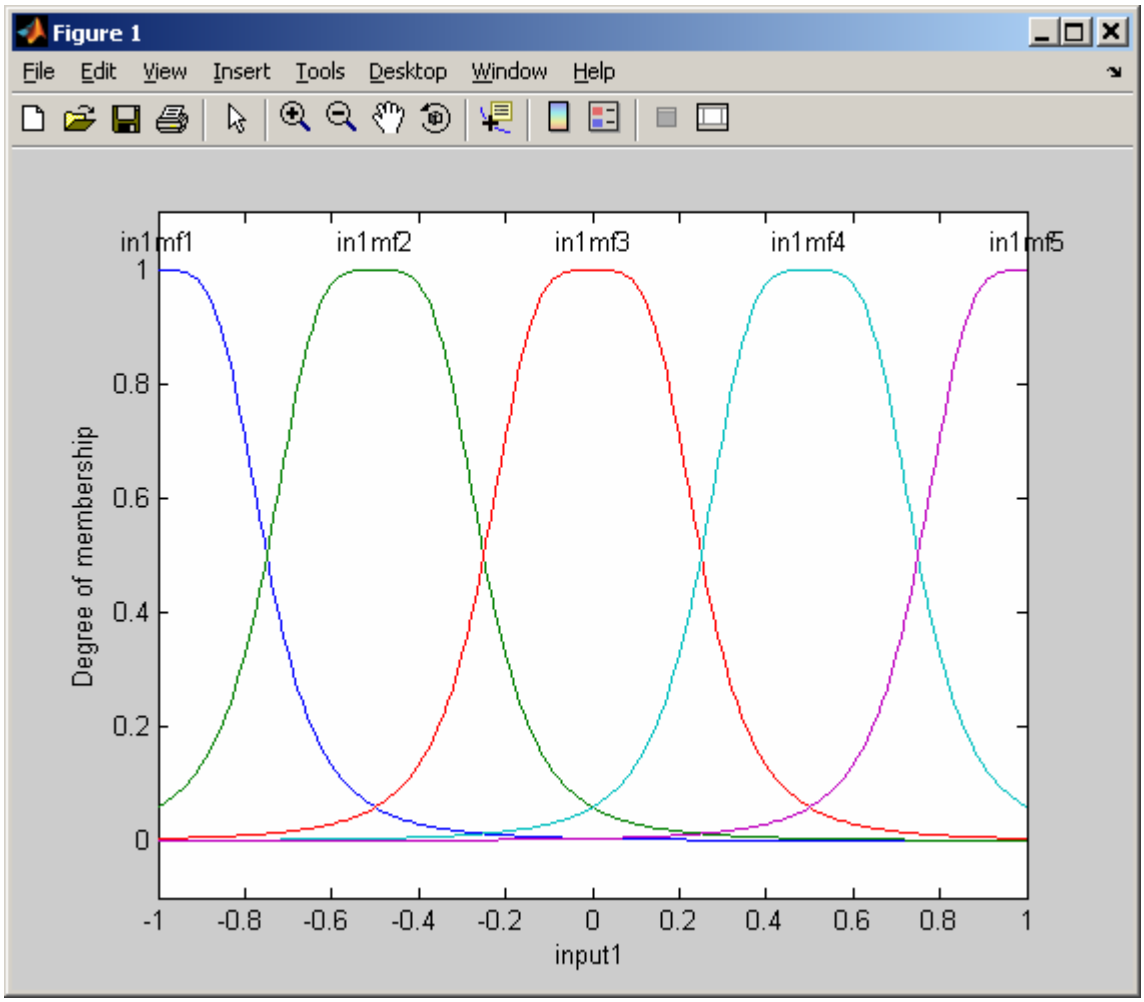
anfis erhält das Ergebnis von genfis1 für den Start der Optimierung.

Bsp. 1:

```
%
% Ein Fuzzy-System soll zur Beschreibung von drei
% zusammengeführten Sinus-Funktionen angepaßt werden
% Die Lösung erfolgt mit Hilfe von MATLAB und der Fuzzy-Logic
% Toolbox
%
% Anzahl der Eingabedatenpunkte und Eingabedaten x
numPts = 51; x = linspace(-1,1,numPts)';
% Ausgabedaten y
y = 0.6*sin(pi*x)+0.3*sin(3*pi*x)+0.1*sin(5*pi*x);
% Speichere die Daten in data, benutze einen Teil fuer Training
% und Prüfung
data = [x y]; %vollständige Datenmenge
trndata = data(1:2:numPts,:); % Trainingsdatenmenge
chkdata= data(2:2:numPts,:);  % Prüfungsdatenmange
% Zeichnen Trainingsdaten (o) und Prüfdaten (x)
plot(trndata(:,1),trndata(:,2),'o',chkdata(:,1),chkdata(:,2),'x')
grid
title('Training data(x) und checking data(o)')
xlabel('x'); ylabel('Measured data for y');
```

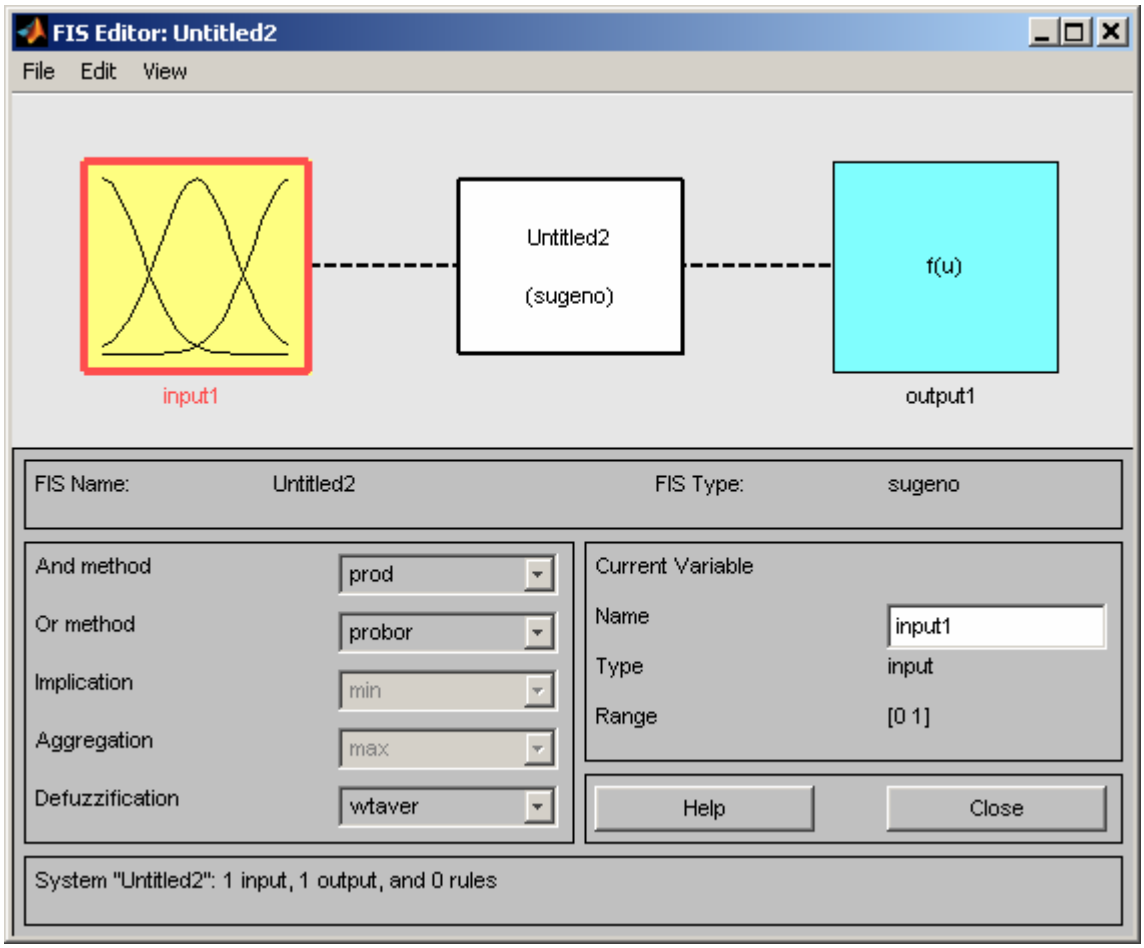



```
% Anwendung von genfis1 zum Erzeugen einer Menge von
% Zugehoerigkeitsfunktionen
nummfs = 5; % Anzahl der Zugehörigkeitsfunktionen
mftype = 'gbellmf'; % Zugehoerigkeitsfunktionstyp
fismat = genfis1(trndata,nummfs,mftype);
plotmf(fismat,'input',1);
```



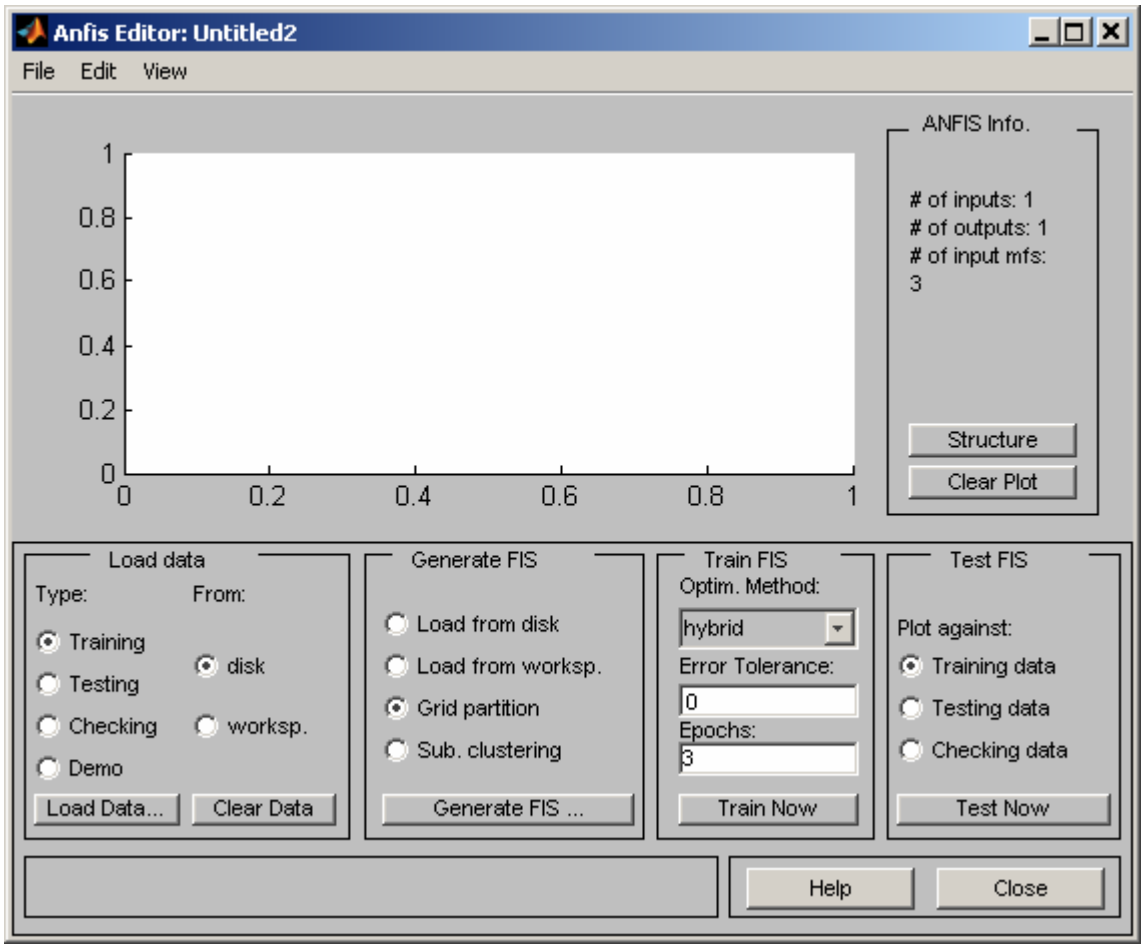
Eine andere Möglichkeit auf dem Weg zur Lösung ist die Verwendung von “Anfis GUI”:

1. Öffne die GUI für Fuzzy Logik mit dem Kommando `fuzzy`, dann wähle File->NEWFIS…->Sugeno

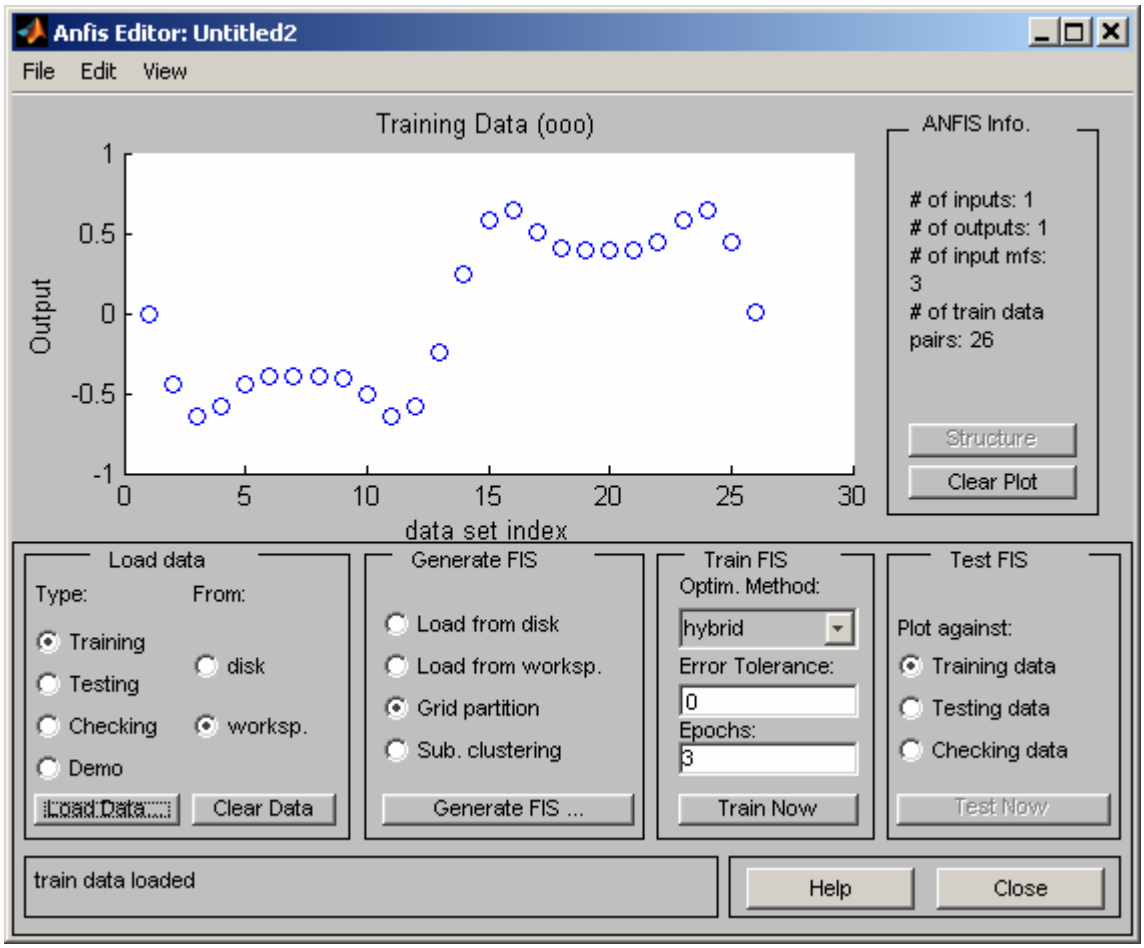


Im Edit-Menü wähle `Anfis`...
Das führt zu der folgenden Ausgabe, die in 4 Teile gegliedert ist:

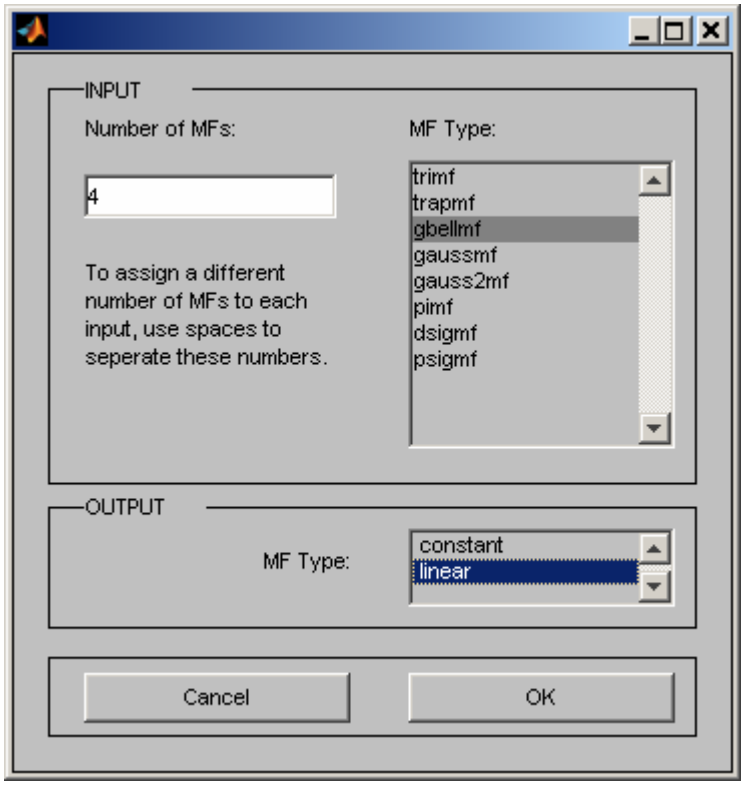
1. Load data
2. General FIS
3. Train FIS
4. Test FIS



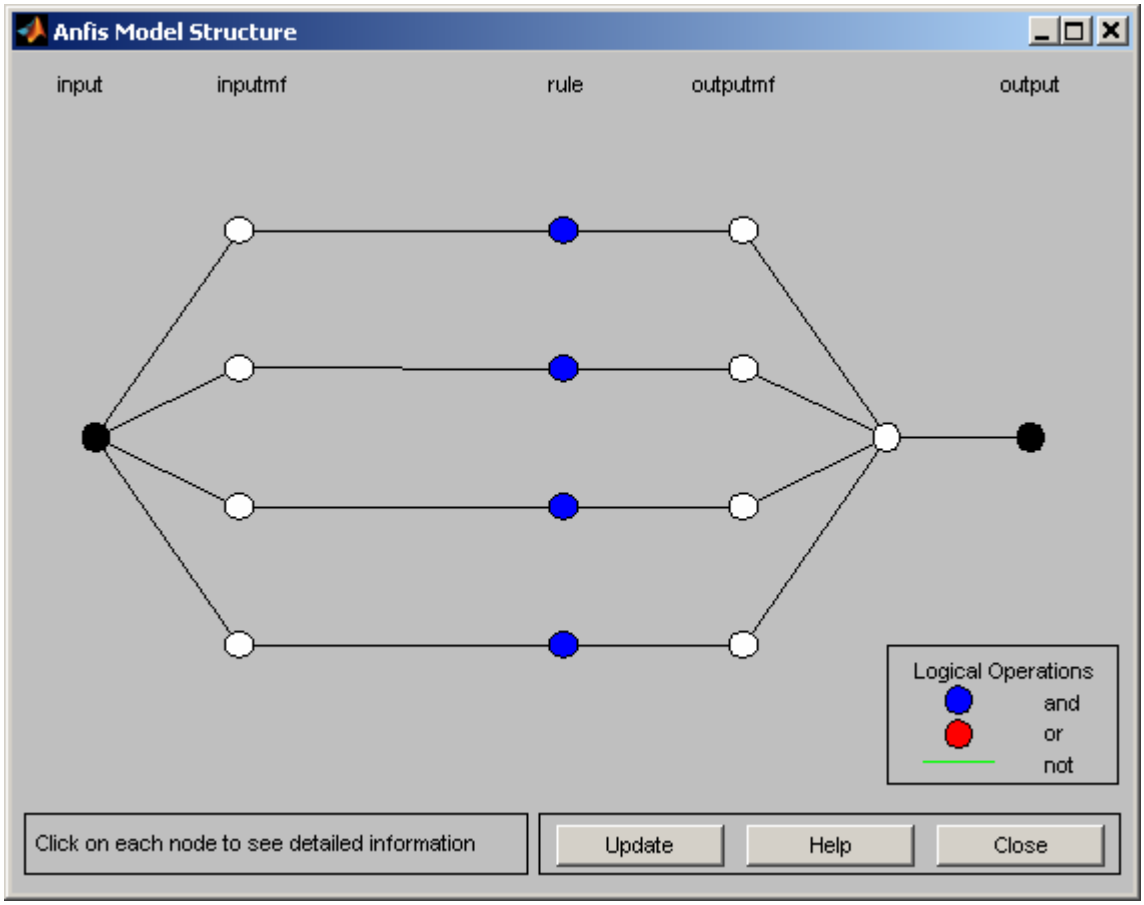
1. Load data
Hier kann der Typ der Daten gewählt werden, z.B.: Daten für Training (default), Test, Prüfung, Demo. Im vorliegenden Beispiel wird mit Trainigsdaten begonnen, die mit MATLAB erstellt wurden. Sie sollten in Form einer Matrix vorliegen, die erste Spalte besteht aus Eingabedaten, die zweite Spalte aus Ausgabedaten. Daten können auf „disk“ oder im „workspace“ abgelegt sein. Hier liegen die Daten im „workspace“. Nach einem Klick auf „Load Data…“ kann „Generate FIS…“ aktiviert werden.
2. Generate FIS
Ein Fuzzy-System kann nun über einen Klick auf „Generate FIS…“ initialisiert werden. Der Radio-Button wird zuvor auf „Grid partition“ eingestellt.



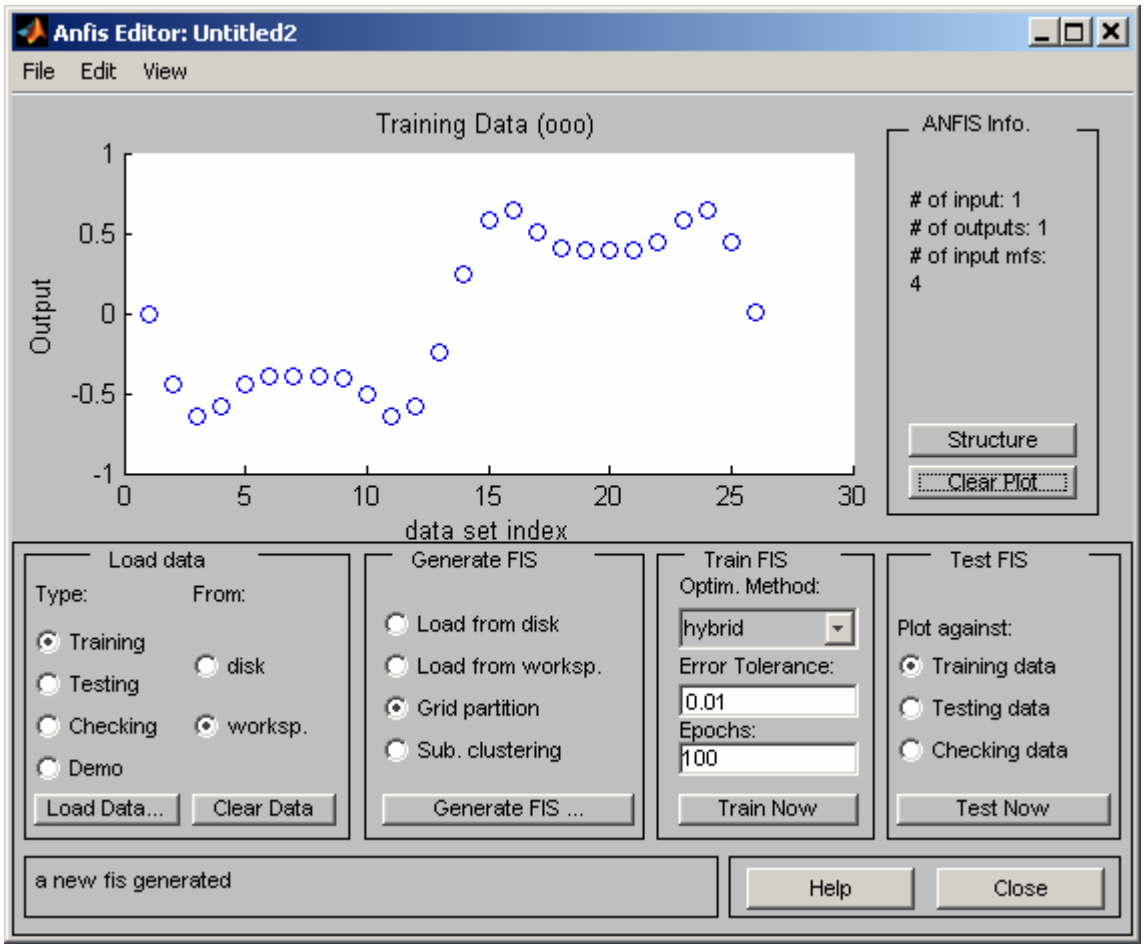
Es folgt die Wahl der Zugehörigkeitsfunktionen.



Die Anfis-Modell-Struktur kann betrachtet werden:

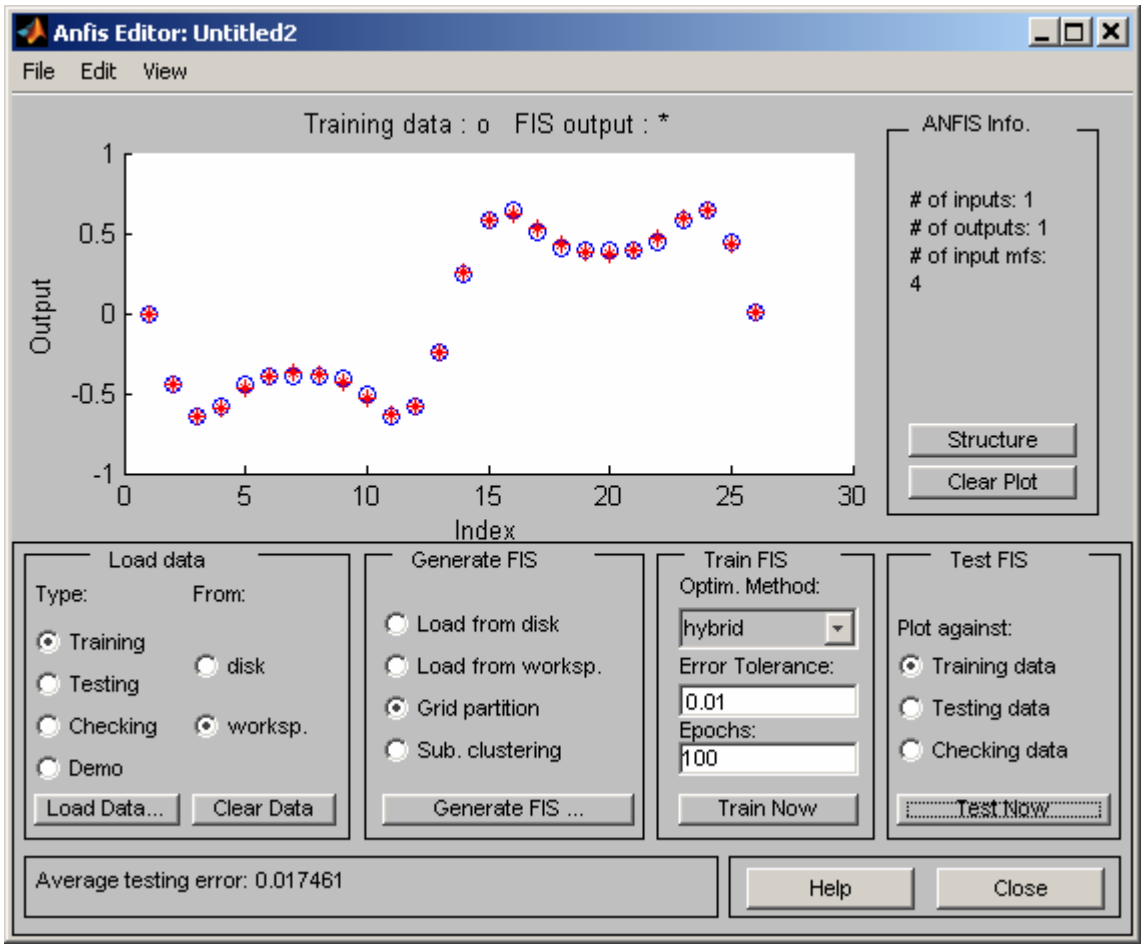


3. Train FIS
Hier erfolgt der wesentliche Teil der Rechnung. Benutzt wird die Hybrid-Regel. Die Parameter werden, wie die folgende Abb. zeigt, eingestellt.



4. Test FIS

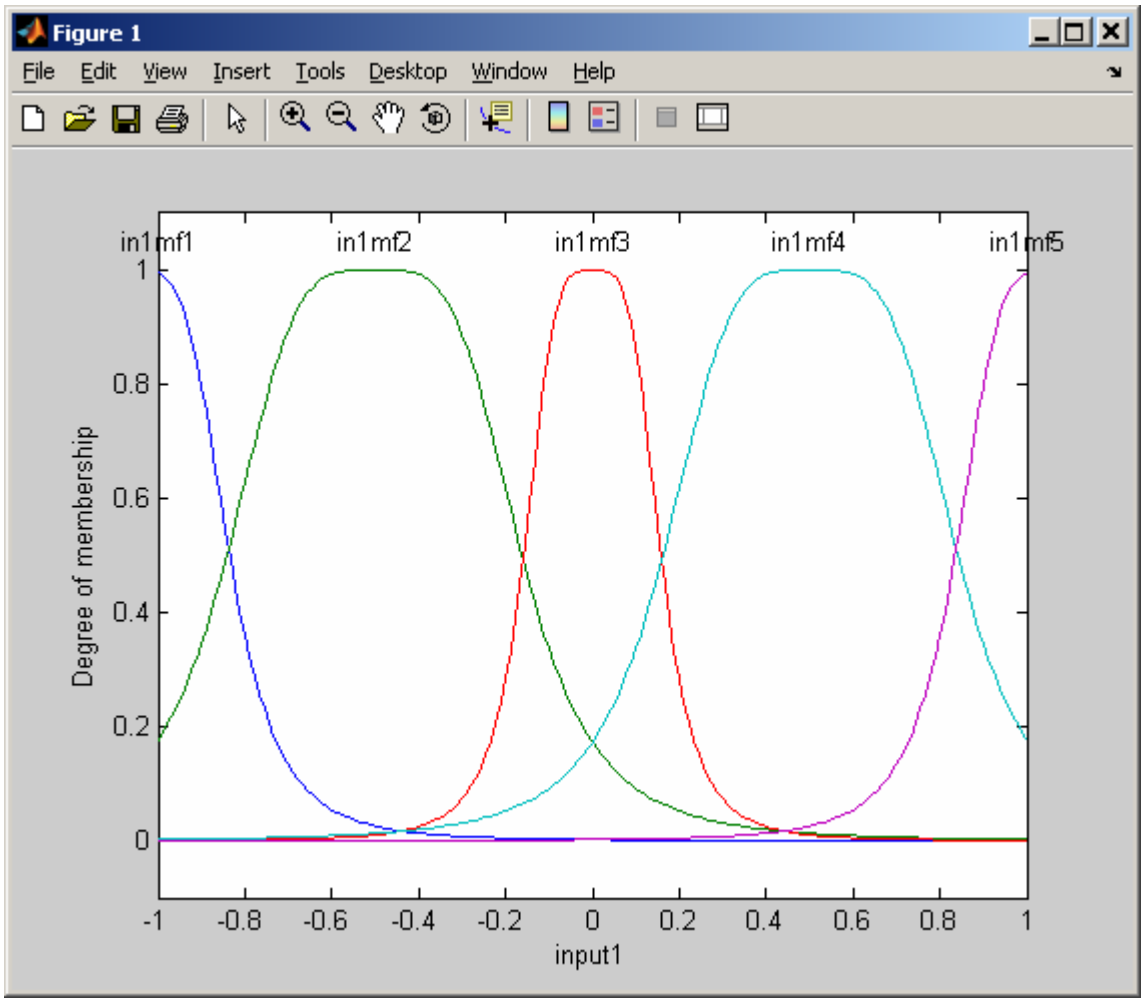
Falls das Ergebnis befriedigend ausfällt, kann ein Test ausgeführt werden. Hier ergibt sich



Auf der MATLAB-Kommando-Seite kann die Optimierungsprozedur durch den Aufruf von „**anfis**“ gestartet werden.

```
% Start der Optimierungsprozedur durch den Aufruf von anfis
% Bestimme die Anzahl der Trainingsschritte
numepochs = 40;
% Start der Optimierung
[fismat1,trnerr,ss,fismat2,chkerr]=...
    anfis(trndata,fismat,numepochs,NaN,chkdata);

ANFIS info:
      Number of nodes: 24
      Number of linear parameters: 10
      Number of nonlinear parameters: 15
      Total number of parameters: 25
      Number of training data pairs: 26
      Number of checking data pairs: 25
      Number of fuzzy rules: 5


Start training ANFIS ...

   1   0.0601596   0.058193
   2   0.0571745   0.0553133
   3   0.0541812   0.0524169
   4   0.0511445   0.0494663
   5   0.0480347   0.0464298
Step size increases to 0.011000 after epoch 5.
   6   ...

Designated epoch number reached --> ANFIS training completed at epoch 40.
```

```
% Darstellung der Zugehoerigkeitsfunktionen
plotmf(fismat1,'input',1);
```




```
out=evalfis(chkdata(:,1),fismat1); %Evaluierung der Ausgabe vom Fuzzy
System
hold; plot(chkdata(:,1),out);      % Zeichnet out vs x
plot(chkdata(:,1),chkdata(:,2));     % Zeichnet y vs x (Trainingsdaten)
plot(x,y);
grid;
```

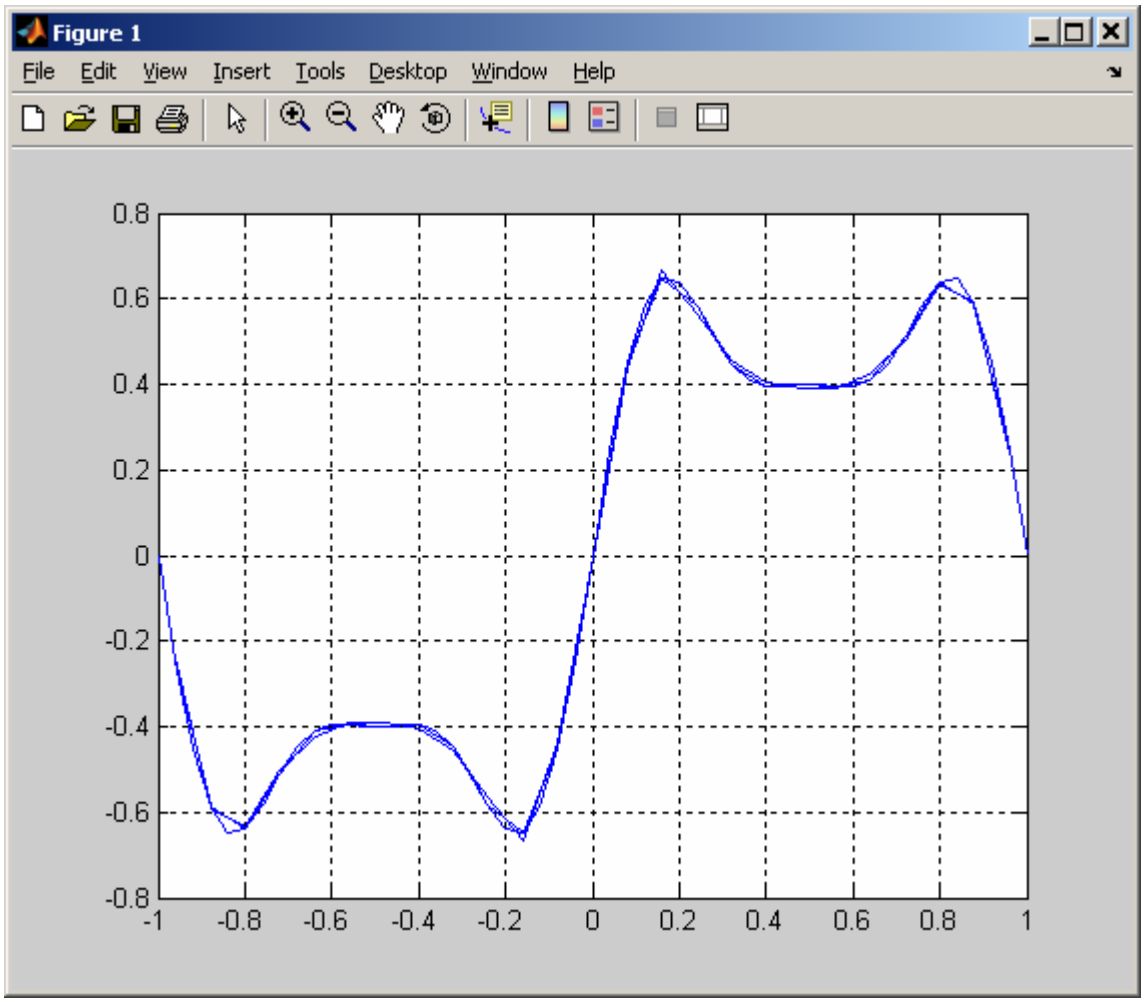


Figure 1
File Edit View Insert Tools Desktop Window Help
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1

3. NEFCON

3.1 Fuzzy Perzeptron[9]

Eine Möglichkeit, ein Modell für ein Neuro-Fuzzy-System zu erzeugen, beruht auf der Idee eines Fuzzy-Perzeptrons. Dabei werden Gewichte auf den Verbindungen durch Fuzzy-Mengen ersetzt (Fuzzy-Gewichte) und Aktivierungs- und Propagierungsfunktionen so modifiziert, dass sie mit Fuzzy-Mengen operieren können. Die Eingabeeinheiten repräsentieren die Eingangsgrößen, die inneren Einheiten stehen für die Fuzzy-Regeln, und die Ausgabeeinheiten stellen die Stellgrößen dar. Die Propagierungsfunktionen zwischen Eingabeschicht und innerer Schicht bestimmen die Zugehörigkeitsgrade der Eingaben zu den Fuzzy-Gewichten der entsprechenden Verbindungen. Durch eine Minimumbildung bestimmt die Aktivierungsfunktion einer inneren Einheit daraus den Erfüllungsgrad einer Regel. Die Propagierungsfunktion zwischen innerer Schicht und Ausgabeschicht verknüpft daraufhin die Erfüllungsgrade mit den Fuzzy-Gewichten der zur Ausgabe führenden Verbindungen. Die Aktivierungsfunktion einer Ausgabeeinheit fasst die ankommenden modifizierten Fuyy-Mengen zu einer zusammen und führt anschließend eine Defuzzifizierung durch.

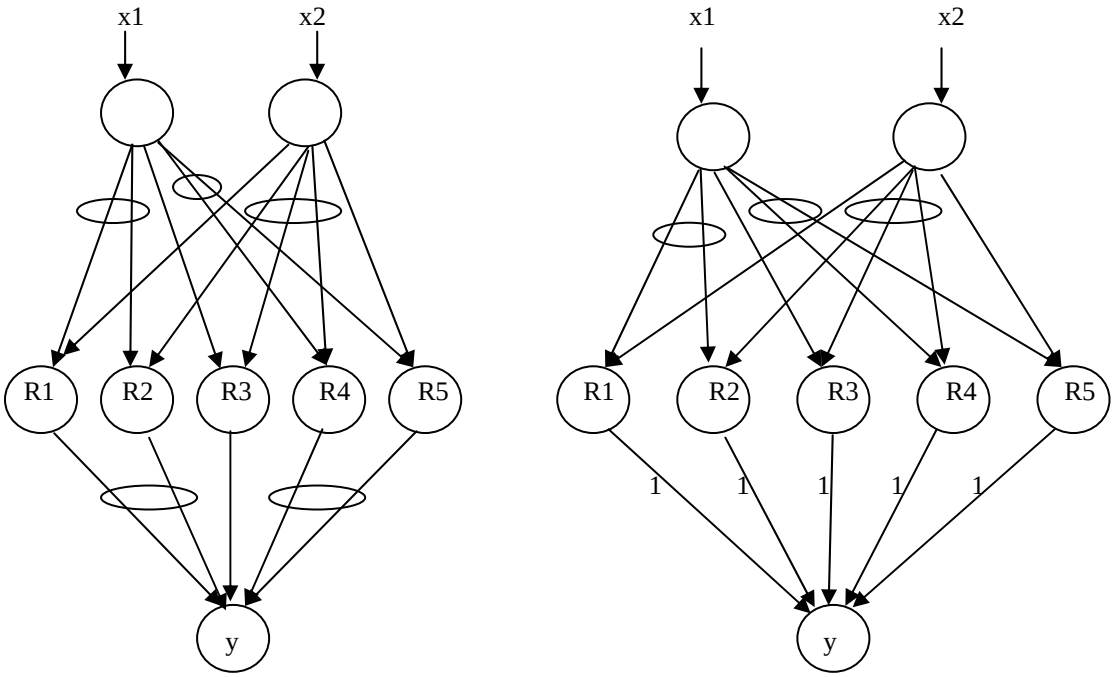


Abb.: Zwei von einem Fuzzy-Perzeptron abgeleitete Neuro-Fuzzy-Systeme. Links ein NEFCON-System für regelungstechnische Aufgaben mit Fuzzy-Gewichten in beiden Verbindungsebenen, rechts ein NEFCLASS-System zur Klassifikation von Daten mit Fuzzy-Gewichten nur un der ersten Verbindungsebene. Die Ringe um die Verbindungen stellen gekoppelte Gewichte das. Diese sind zu jeder Zeit identisch.

3.2 Erlernen der Fuzzy Regeln

Für das Erlernen der Fuzzy-Regeln stehen 2 Methoden zur Verfügung:

1. Man beginnt ohne Regeln und fügt eine Regel nach der anderen hinzu bis das System eine zufrieden stellende Leistung erbringt
2. Man beginnt mit einer Basis aus allen möglichen erzeugbaren Regeln und streicht nicht benötigte nacheinander wieder heraus.

[9] abgeleitet von einem dreischichtigen vorwärtsgetriebenen Neuronalen Netz (Multilayer Perzeptron)

Methode 2 ist für online lernende Modelle geeignet. Die Komplexität aus solchen Systemen steigt jedoch rasch mit der Anzahl der Ein- und Ausgangsvariablen und der Anzahl der Fuzzy-Mengen, da für jede Kombination eine Regel generiert werden muß.
Bsp.: 2 Eingangsvariable, eine Ausgangsvariable und eine Fuzzy-Partionierung von 8 Fuzzy Mengen zu jeder Variablen führen bereits zu 512 Regeln. Erhöht man die Zahl der Eingangsvariable auf 4, so sind es bereits 32768 Regeln.
Die Komplexität des Systems steigt somit exponentiell mit der Zahl der Freiheitsgrade.
Der Lernalgorithmus basiert auf der Idee eines Fuzzy-Fehlers und entspricht dem verstärkenden Lernen. Es ist in der Lage, Fuzzy-Mengen zur Repräsentation linguistischer Terme und linguistischer Regeln adaptiv zu verändern

3.3 Fuzzy-Fehler-Propagierung

Das NEFCON-Modell benutzt folgende Variante des Backpropagation-Verfahrens. Es lernt durch „verstärkendes Lernen“ und verwendet dazu einen Fuzzy-Fehler.
Der benutzer gibt eine Reihe von Regeln, die beschreiben, wann die Leistung des Systems gut oder schlecht ist, z.B. „Wenn die Temperatur angenehm ist, dann ist die Leistung hoch“. Diese Regeln werden wie die eines Fuzzy-Reglers ausgewertet und zur Auswertung der Systemparameter (der Fuzzy-Mengen) herangezogen. Der Fuzzy-Fehler wird dabei als (negatives) Verstärkungssignal genutzt und wie beim Backpropagation-Algorithmus (beginnend bei der Ausgabeinheit) rückwärts durch das Netzwerk propagiert. Dabei wird es von den Regeleinheiten lokal zur Adaption der Fuzzy-Mengen genutzt. Für jede Zugehörigkeitsfunktion muß ermittelt werden, ob und wie sie zu verändern ist. Es werden nur solche Zugehörigkeitsfunktionen verändert, die mit Regeleinheiten in Verbindung stehen, deren Aktivierung ungleich Null ist.
Zur Realisierung von verstärkendem Lernen[10] muß entschieden werden, ob eine Regel für ihre Reaktion auf den aktuellen Zustand zu belohnen oder zu bestrafen ist. Mit der Belohnung soll erreicht werden, dass in der gleichen Situation die Regel einen stärkeren Beitrag zur Regelaktion leistet und sie somit positiv beeinflusst. Eine Bestrafung soll den Beitrag einer Regel abschwächen, um die Regelaktion weniger negativ zu beeinflussen.

3.4 das NEFCLASS-Modell

Dieses Modell ist ebenfalls vom Fuzzy-Perzeptron abgeleitet, verzichtet aber auf Fuzzy-Gewichte in der 2. Verbindungsebene. Die in diesem Modell eingesetzten Fuzzy-Regeln sind von der Form:

IF x_1 IS A_1 AND … AND x_n IS A_n THEN class C1
A_i: linguistische Variable

Die Konklusion der Regeln besteht nur aus einer Klassenzuteilung. NEFCLASS lernt eine Regelbasis durch sukzessives Hinzufügen von Regeln, d.h. von Knoten der inneren Schicht. Anschließend werden die verschiedenen Fuzzy-Mengen optimiert. Idealerweise erhält man einen Klassifikator, der in Form von linguistischen Regeln interpretiert werden kann.

[10] vgl. Skriptum 3.2.3.4.1