h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi
FACHBEREICH INFORMATIK

Hochschule Darmstadt
Fachbereich Informatik

Neuro-Fuzzy-Systeme

Computational Intelligence
Dr. Norbert Waleschkowski

von

Walter Noll	Martikel-Nr.: 546014
Stefan Sonski	Martikel-Nr.: 714352
Sven Eisenhauer	Martikel-Nr.: 707173

27. Juni 2011

Inhaltsverzeichnis

1 Einleitung

Bei Neuro-Fuzzy-Systemen handelt es sich um die Kombination eines Neuronalen Netzes mit einem Fuzzy-System. Da der Umfang dieser Arbeit beschränkt ist, setzen die Autoren bestimmte Vorkenntnisse aus den Bereichen Neuronale Netze und Fuzzy-Logik voraus. Dazu zählen der grundlegende Aufbau eines mehrschichtigen Perzeptrons und der Ablauf des Backpropagation-Algorithmus aus dem Bereich der Neuronalen Netze. Die Kenntnis der Begriffe T-Norm und T-Conorm, Zugehörigkeitsfunktion, linguistische Variable sowie Mamdani- und Sugeno-Controller bildet weitere Voraussetzung. Darin eingeschlossen sind Kenntnisse über Fuzzy-Mengen und -Regeln sowie Fuzzy-Logik im Allgemeinen.
Basierend auf diesem Grundwissen betrachten wir bestimmte Eigenschaften von Fuzzy-Systemen und Neuronalen Netzen.
Im Allgemeinen orientiert sich der Entwurf von Fuzzy-Controllern an Expertenwissen, das der Fuzzy-Controller nachahmt. Dies setzt voraus, dass für das zu regelnde System die Reaktionen und das Wissen des Experten bekannt sind. Ohne dieses Wissen lassen sich weder Fuzzy-Mengen noch Fuzzy-Regeln entwerfen. Ebenso besteht das Problem der Vollständigkeit. Da ein Fuzzy-Controller nicht zur Laufzeit lernen kann, muss die Lösung des Problems beim Entwurf vollständig und richtig über alle Eingabedaten sein. Auch kann sich der Fuzzy-Controller nicht an sich ändernde Parameter z. B. durch mechanischen Verschleiß anpassen.
Im Gegensatz dazu stehen Neuronale Netze, deren Hauptmerkmal ihre Lernfähigkeit darstellt. Als Nachteil steht dem gegenüber, dass evtl. vorhandenes Wissen über das Problem nicht in einem Neuronalen Netz implementiert werden kann. Gleichzeitig besteht darin der Vorteil, dass eine Problemlösung durch rein daten-getriebenes Lernen erfolgt. Der Lernerfolg eines Neuronalen Netzes stellt einen Unsicherheitsfaktor dar, da nicht vorher bestimmt werden kann, ob und wann sich ein Trainingserfolg einstellt. Unter ungünstigen Voraussetzung kann eine Überanpassung an die Trainings- oder Validierungsdaten eintreten, was die Lösung für reale Daten negativ beeinflussen kann. Da sich Neuronale Netze wie eine *Blackbox* nach außen darstellen, lässt sich ihr innerer Zustand nicht oder nur sehr eingeschränkt interpretieren.
Die Motivation zur Kombination Neuronaler Netze und Fuzzy-Systeme zu Neuro-Fuzzy-Systemen liegt in der Idee des *best of both worlds*. Das Ziel besteht hierbei darin, mit einer Stärke des einen Ansatzes in einem bestimmten Bereich eine Schwäche des anderen in diesem Bereich zu kompensieren, um so zu einem leistungsfähigeren Gesamtsystem zu kommen.

2 Grundlagen

Bei der Betrachtung von Neuro-Fuzzy-Systemen liegt das Hauptunterscheidungsmerkmal im Grad der Kopplung von Neuronalem Netz und Fuzzy-System. Kooperative Neuro-Fuzzy-Systemezeichnen sich durch eine lose Kopplung aus, wobei hybride Neuro-Fuzzy-Systemeeine enge Kopplung aufweisen.

2.1 Kooperative Neuro-Fuzzy-Systeme

Bei kooperativen Neuro-Fuzzy-Systemen bleibt die Eigenständigkeit des beteiligten Neuronalen Netzes und Fuzzy-Systems erhalten und es besteht eine klar definierte Trennung. Die einfachste Kombination eines Neuronalen Netzes und eines Fuzzy-Systems besteht in dem Ansatz der Aufbereitung der Eingabedaten des Fuzzy-Controllers durch das Neuronale Netz. Ebenso lassen sich die Ausgaben eines Fuzzy-Controllers durch ein Neuronales Netz aufbereiten. Hierbei beeinflussen sich die beiden Systeme gegenseitig nicht sondern nur die Daten.

Wenn sich die Fuzzy-Mengen der linguistischen Variablen eines Fuzzy-Controllers zur Regelung eines Systems nicht bestimmen lassen, aber gleichzeitig Datenpaare aus Eingabe und gewünschter Ausgabe vorliegen, lassen sich diese Fuzzy-Mengen mit einem Neuronalen Netz erzeugen. Dies erfolgt offline durch Bestimmung geeigneter Parameter der Zugehörigkeitsfunktionen oder Approximation der Zugehörigkeitsfunktionen durch das Neuronale Netz. Die Abbildung 1 zeigt den schematischen Aufbau einer solchen Lösung. Das Training und die Erzeugung der Fuzzy-Mengen erfolgt offline, so dass das Neuronale Netz zur Laufzeit nicht benötigt wird.

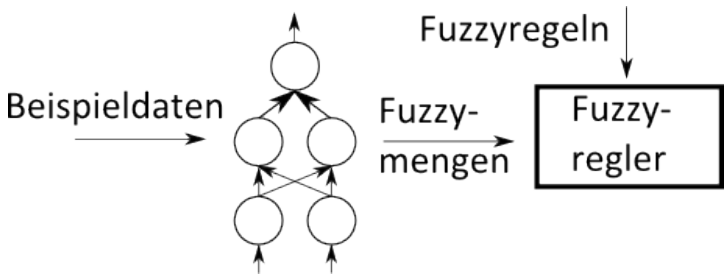

Abbildung 1: Neuronales Netz zum Offline-Lernen der Fuzzy-Mengen eines Fuzzy-Controllers

Mit einem ähnlichen Ansatz lassen sich auch Fuzzy-Regeln erzeugen, wie Abbildung 2 zeigt. [BKKN03] nennt hierzu Clustering-Verfahren oder selbstorganisierende Karten als Neuronale Methode zur Bestimmung der Fuzzy-Regeln. Auch hier arbeitet das Neuronale Netz offline.

Die Anpassung der Fuzzy-Mengen eines Fuzzy-Controllers durch ein Neuronales Netz kann auch online erfolgen, wie in Abbildung 3 gezeigt. Dazu benötigt das Neuronale Netz die Ausgaben des Fuzzy-Controllers als Eingabe zur Fehlerbestimmung.

Mit einer solchen Fehlerbestimmung lässt sich auch ein System wie in Abblidung 4 gezeigt entwerfen, das die Gewichte der Fuzzy-Regeln des Fuzzy-Controllers online verändert.

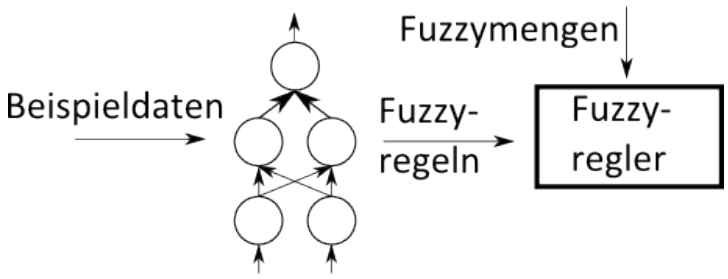

Abbildung 2: Neuronales Netz zum Offline-Lernen der Fuzzy-Regeln eines Fuzzy-Controllers

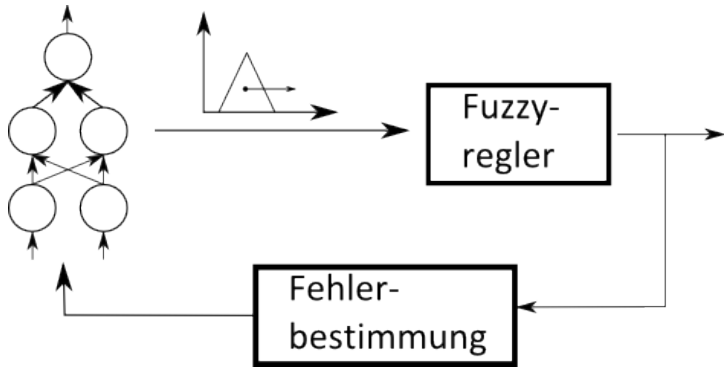

Abbildung 3: Neuronales Netz zur Online-Anpassung der Fuzzy-Mengen eines Fuzzy-Controllers

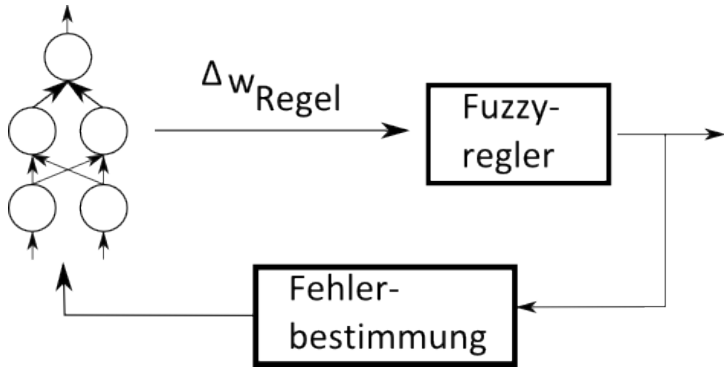

Abbildung 4: Neuronales Netz zur Online-Anpassung der Fuzzy-Regeln eines Fuzzy-Controllers

2.2 Hybride Neuro-Fuzzy-Systeme

Bei hybriden Neuro-Fuzzy-Systemen verschmelzen Neuronales Netz und Fuzzy-Controller zu einem System bei dem keine klare Trennung zwischen Neuronalem und Fuzzy-Teil

mehr möglich ist. Abbildung 5 zeigt das Schema eines solchen hybriden Neuro-Fuzzy-Controllers nach [BKKN03].

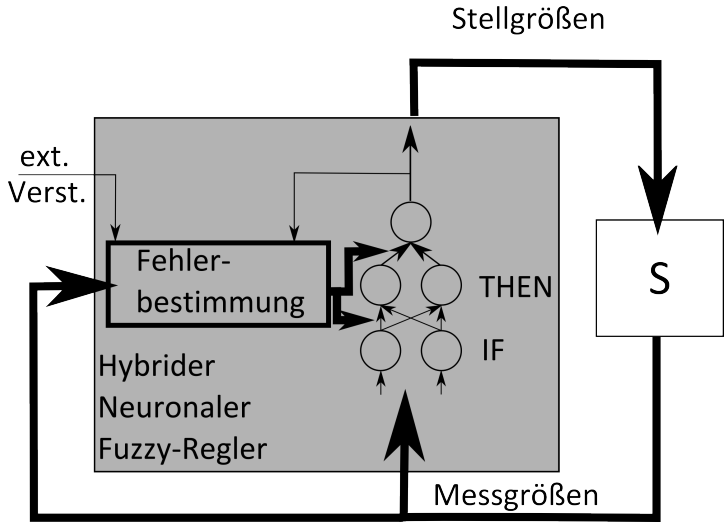

Abbildung 5: Schema eines hybriden Neuro-Fuzzy-Controllers

3 Das generische Fuzzy-Perzeptron

Als ein Beispiel hybrider Neuro-Fuzzy-Systemebetrachten wir in diesem Abschnitt das generische Fuzzy-Perzeptronnach [BKKN03] genauer. Abbildung 6 zeigt ein generisches Fuzzy-Perzeptron zur Lösung des XOR-Problems. Dabei handelt es sich um ein dreischichtiges vorwärtsgetriebenes Neuronales Netz bestehend aus einer Eingabeschicht, einer versteckten Schicht und einer Ausgabeschicht. Die Gewichte der Verbindungen zwischen den Neuronen werden durch den Zugehörigkeitsgrad der Ausgabe der vorherigen Schicht zu den beiden abgebildeten Fuzzy-Mengen s für small und b für big repräsentiert. Die Ausgabefunktion der Neuronen der versteckten Schicht bildet eine T-Norm, die Ausgabefunktion der Ausgabeschicht bildet die entsprechende T-Conorm, hier min und max.

Wertetabelle Die Tabelle 1 zeigt die Ausgaben der jeweiligen Schichten für die möglichen Eingaben des XOR-Problems. An den Ausgabewerten der Ausgabeschicht lässt sich die korrekte Lösung des XOR-Problems durch das generische Fuzzy-Perzeptronerkennen.

3.1 Beschreibung

Dieser Abschnitt beschreibt die einzelnen Komponenten des generischen Fuzzy-Perzeptrons genauer.

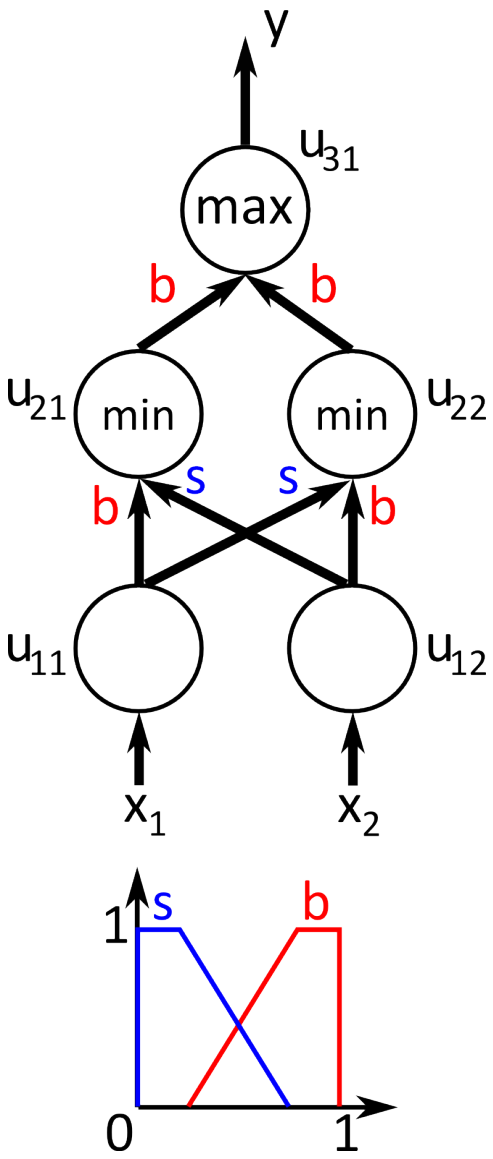

Abbildung 6: Generisches Fuzzy-Perzeptron zur Lösung des XOR-Problems

Schichten Die drei Schichten des generischen Fuzzy-Perzeptrons lassen sich jeweils als Menge von Neuronen beschreiben.

- Eingabeschicht $U_1 = \{x_1, \ldots, x_n\}$
- Versteckte Schicht $U_2 = \{R_1, \ldots, R_r\}$
- Ausgabeschicht $U_3 = \{y_1, \ldots, y_m\}$

Jedes Neuron verfügt dabei über drei Funktionen:

- Die Netzeingabefunktion net_u

x_1	x_2	$out_{u_{11}}$	$out_{u_{12}}$	$out_{u_{21}}$	$out_{u_{22}}$	$out_{u_{31}}$ = y
0	0	0	0	min(0,1)= 0	min(0,1)= 0	max(0,0)= 0
0	1	0	1	min(0,0)= 0	min(1,1)= 1	max(0,1)= 1
1	0	1	0	min(1,1)= 1	min(0,0)= 0	max(1,0)= 1
1	1	1	1	min(1,0)= 0	min(1,0)= 0	max(0,0)= 0

Tabelle 1: Ausgaben der Neuronen für das XOR-Problem

- Die Aktivierungsfunktion act_u
- Die Ausgabefunktion out_u

Die Verarbeitung je Neuron u erfolgt dabei in der Reihenfolge $net_u \rightarrow act_u \rightarrow out_u$.

Gewichte Das Gewicht der Verbindung zwischen zwei Neuronen unterschiedlicher Schichten ist definiert durch die Fuzzy-Menge μ für die Verbindungen zwischen Neuronen der Eingabeschicht und Neuronen der versteckten Schicht. Für die Verbindungen zwischen Neuronen der versteckten Schicht und der Ausgabeschicht erhalten ihr Gewicht über die Fuzzy-Menge ν. Das Gewicht w der Verbindung zwischen dem Neuron u und dem Neuron v der nachgelagerten Schicht lässt sich also wie folgt berechnen.

$$w_{vu} = \begin{cases} \mu_j^{(i)} & \text{für } u = x_i, v = R_j \\ \nu_j^{(k)} & \text{für } u = R_j, v = y_k \\ \text{undefiniert} & \text{sonst} \end{cases}$$

Mit $1 \leq i \leq n$, $1 \leq j \leq r$, $1 \leq k \leq m$
$\mu_j^{(i)} \in F(\mathbb{R})$, $\nu_j^{(k)} \in F(\mathbb{R})$
und $F(\mathbb{R}) :=$ die Menge aller Fuzzy-Mengen über $\mathbb{R}$

Eingabeschicht Für Neuronen der Eingabeschicht $u \in U_1$ lassen sich die drei Funktionen net_u, akt_u und out_u wie folgt beschreiben.

Die Netzeingabefunktion net_u stellt eine Abbildung von $\mathbb{R}$ auf $\mathbb{R}$ für die externe Eingabe ext_u dar.
$f_{net}^{(u)} : net_u = ext_u$

Die Aktivierungsfunktion act_u stellt eine Abbildung der Netzeingabe net_u von $\mathbb{R}$ auf $\mathbb{R}$ dar.
$f_{act}^{(u)} : act_u = net_u$

Die Ausgabefunktion out_u bildet die Aktivierung act_u von $\mathbb{R}$ auf $\mathbb{R}$ ab.
$f_{out}^{(u)} : out_u = act_u$

Versteckte Schicht Für Neuronen der versteckten Schicht $u \in U_2$ lassen sich die drei Funktionen net_u, akt_u und out_u wie folgt beschreiben.

Die Netzeingabefunktion net_u stellt eine Abbildung von $(\mathbb{R} \times F(\mathbb{R}))$ auf $[0,1] \in \mathbb{R}$. Sie verwendet dabei das Gewicht $w_{uu'}$ zwischen u und $u' \in U_1$ unter Bildung der T-Norm. Dieses Beispiel verwendet hierfür die Minimum-Norm min.

Die Aktivierungsfunktion act_u stellt eine Abbildung von $\mathbb{R}$ auf $\mathbb{R}$ für die Netzeingabe net_u dar.
$f_{act}^{(u)} : act_u = net_u$

Die Ausgabefunktion out_u bildet die Aktivierung act_u von $\mathbb{R}$ auf $\mathbb{R}$ ab.
$f_{out}^{(u)} : out_u = act_u$

Ausgabeschicht Für Neuronen der $u \in U_3$ lassen sich die drei Funktionen net_u, akt_u und out_u wie folgt beschreiben.

Die Netzeingabefunktion net_u verwendet die T-Conorm über die Ausgaben aller Neuronen der versteckten Schichten $u' \in U_2$ mit dem Gewicht $w_{uu'}$ zur Berechnung des Netzeingabewertes. Dieses Beispiel verwendet die Maximum-Conorm max.

Die Aktivierungsfunktion act_u stellt eine Abbildung von $F(\mathbb{R})$ auf $F(\mathbb{R})$ für die Netzeingabe net_u dar.
$f_{act}^{(u)} : act_u = net_u$

Die Ausgabefunktion out_u Durch eine geeignete Abbildung der Aktivierung act_u von $F(\mathbb{R})$ auf $\mathbb{R}$ erfolgt hier die Defuzzifizierung der Aktivierung act_u zu einem scharfen Wert.

3.2 Fuzzy-Fehlerbestimmung

Für ein Lernverfahren bildet die Bestimmung des Fehlers für eine bestimmte Eingabe bei gewünschter Ausgabe ein zentrales Element. Für ein Neuron u der Ausgabeschicht lässt sich der Fehler E für ein Muster p nach [BKKN03] wie folgt berechnen.

$$E_u^{(p)} = 1 - \exp\left(\beta\left(\frac{t_u^{(p)} - o_u^{(p)}}{range_u}\right)^2\right)$$

Unter Verwendung von

- $u \in U_3$ bezeichnet das Neuron der Ausgabeschicht.
- Der Zielwert t bezeichnet die gewünschte Ausgabe für das Muster p.
- Der Ausgabewert o repräsentiert die aktuelle Ausgabe des Neurons.

- $range_u$ bezeichnet die Differenz zwischen dem maximalem und dem minimalem Ausgabewert des Neurons u.
- β bildet einen multiplikativen Toleranzfaktor.

Aus dieser Formel ist leicht zu erkennen, dass es sich hierbei um ein Delta-Lernverfahren handelt. Stimmen Ausgabe und Zielwert überein, wird der Exponent im zweiten Term 0. Durch $e^0 = 1$ ergibt sich wie erwartet ein Fehler $E = 1 - 1 = 0$

3.3 Fuzzy-Backpropagation

Der Backpropagation-Algorithmus für dieses Perzeptron lässt sich nach [BKKN03] wie folgt formulieren.

1. Wähle ein beliebiges Muster p einer gegebenen festen Lernaufgabe und propagiere den Eingabevektor i^p
2. Bestimme
$$\delta_u^{(p)} = \begin{cases} sgn\left(t_u^p - out_u^p\right) * E_u^{(p)} & \text{für } u \in U_3 \\ \sum_{v \in U_3} act_u^{(p)} * \delta_v^{(p)} & \text{für } u \in U_2 \end{cases}$$
3. Bestimme
$$\Delta_p w_{vu} = f\left(\delta_u^{(p)}, act_u^{(p)}, net_u^{(p)}\right)$$

Wiederhole diese Schritte bis der Gesamtfehler

$$E = \sum_p \sum_{u \in U_3} E_u^{(p)}$$

hinreichend klein ist oder ein anderes Abbruchkriterium erfüllt wird.
Im zweiten Schritt lässt sich erkennen, wie der Fehler der Ausgabeneuronen $v \in U_3$ auf die Neuronen der versteckten Schicht $u \in U_2$ wirken.
Die Funktion f, die in Schritt drei Verwendung findet ist nicht genauer spezifiziert. Hier könnte beispielsweise eine Anpassung der Parameter der Zugehörigkeitsfunktionen der Fuzzy-Mengen stattfinden. Bei den Fuzzy-Mengen in Abbildung 6 könnte z. B. die Steigung der Geraden variiert werden.

4 ANFIS

ANFIS wurde als weiteres Modell eines Neuro-Fuzzy-Systems 1992 von Jyh-Shing Roger Jang an der University of California at Berkeley entwickelt.
Es stellt ein Adaptive Network–based Fuzzy Inference System, also ein lernfähiges netzwerkbasiertes Fuzzy-Schlussfolgerungs-System dar.
Hierfür werden Fuzzy-Regeln in einem Neuronalen Netz implementiert.
Kennzeichnend für das ANFIS-Modell ist, daß die jeweiligen Zugehörigkeitsfunktionen festgelegt sind, währen sich die Parameter dieser Funktionen beim Lernvorgang verändern können.

Mit ANFIS können zwei Typen von Fuzzy-Controllern emuliert werden.

- Tsukamoto-Regler
- Sugeno-Takagi-Kang-Regler

Der Mamdani-Regler kann nicht nachgebildet werden, da die Weitergabe von Fuzzymengen mit Neuronen nicht erreicht werden kann. [Sau08]

Als Lernalgorithmus wird für ANFIS ein Hybrid-Lern-Algorithmus verwendet.

4.1 Tsukamoto-Regler

Der Tsukamoto-Regler ist einer der beiden von ANFIS emulierbaren Fuzzy-Controllern. Seine Funktionsweise kann wie folgt beschrieben werden.

- Jede Regel aus der Regelbasis, des Controllers wird mit einer monotonen Zugehörigkeitsfunktion (ZGF) verknüpft. Im Bild unten C1, C2.
- Die Ergebnisse dieser Zugehörigkeitsfunktion sind jeweils ein crisper Wert.
- Der Ausgangswert z des Controllers wird gebildet, in dem das gewichtete Mittel aus den einzelnen crispen Funktionswerten z1, z2, ... zn berechnet wird. Das Ergebnis ist ebenfalls ein crisper Wert. Er muß daher nicht aufwendig defuzzyfiziert werden, wie es zum Beispiel bei einem Mamdani-Controller notwendig ist.

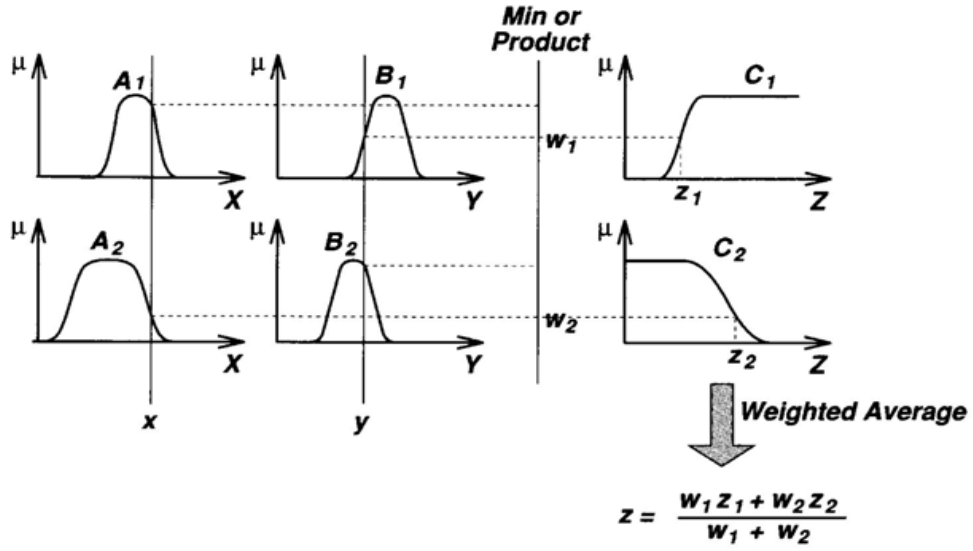

Abbildung 7: Das Tsukamoto-Modell [Bar05]

4.2 Sugeno-Takagi-Kang-Regler

Der Sugeno-Takagi-Kang-Regler ist der zweite, von ANFIS emulierbare Fuzzy-Controller.

Seine Funktionsweise kann wie folgt beschrieben werden.

Eine typische Regel mit zwei Eingangswerten lautet zum Beispiel:

- If x is A1 and y is B1 then z = f(x,y) [Mic02]
  Wobei A1, B1 Werte linguistischer Variablen und x, y, z Crispe Werte sind.

- Die anzuwendenden Regeln b.z.w. Funktionen werden nun anhand der Eingangswerte bestimmt.

- Der Ausgangswert z des Controllers wird ebenfalls gebildet, in dem das gewichtete Mittel aus den einzelnen crispen Funktionswerten z1, z2, ..., zn berechnet wird. Das Ergebnis ist daher wie bei dem zuvor vorgestellten Controller auch ein crisper Wert, welcher nicht aufwendig defuzzyfiziert werden muss, wie es bei einem Mamdani-Controller erforderlich ist.

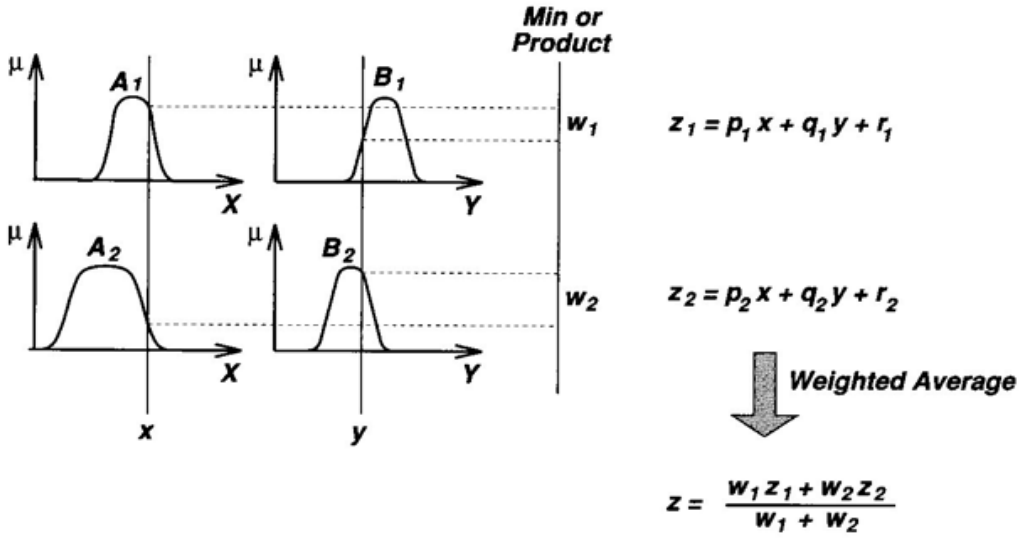

Abbildung 8: Das Sugeno-Takagi-Kang-Modell [Bar05]

Ein Beispiel, zur Verdeutlichung, wie die anzuwendende Regel ermittelt wird:

- Gegeben seien A1 und A2 als crispe Eingangswerte.
- Aufgrund vordefinierter Zugehörigkeitsfunktionen ZGF ergebe sich, daß der gegebene Wert für A1 der ZGF nK und der gegebene Wert für A2 der ZGF pK angehöre.
- So ist in diesem Beispiel aus der nachfolgend aufgeführten Tabelle zu entnehmen, daß die Regel eN anzuwenden ist.
  Diese Regel könnte dann zum Beispiel so aussehen: ZeN = 0,01x + 0,025y + 7

$A_2 \Downarrow A_1 \Rightarrow$	nG	nK	eN	pK	pG
nG	pG	pG	pK	eN	nK
nK	pG	pG	pK	eN	nK
eN	pG	pK	eN	nK	nG
pK	pK	eN	nK	nG	nG
pG	pK	nK	nG	nG	nG

n negativ
p positiv
e etwas
N Null
K Klein
G Groß

Tabelle 2: Bestimmung von Fuzzy-Regeln aufgrund der Zugehörigkeit der Eingangswerte zu bestimmten Fuzzy-Mengen [Bar05]

4.3 ANFIS-Architektur

Die ANFIS Architekturen zur Nachbildung der beiden zuvor vorgestellten Fuzzy-Reglern bestehen jeweils aus einem 5-Schichten Modell.

Wie zu Anfang schon erwähnt werden im ANFIS Modell die Parameter der Regeln erlernt, während die Regeln selbst von Anfang an feststehen. In den nachfolgend gezeigten ANFIS-Architekturen werden die Optimierungen der Parameter in der ersten und vierten Schicht vorgenommen.

Die einzelnen Schichten dieser Architekturen übernehmen nun folgende Aufgaben.
In der ersten Schicht wird die Zugehörigkeit zu den Fuzzymengen betimmt.
Innerhalb der zweiten Schicht werden die Gewichte der Regeln durch Produktbildung erstellt.
Die dritte Schicht dient der Berechnung der normalisierten Gewichte.
Während in der vierten Schicht die Berechnung der Ergebnisse der einzelnen Regeln stattfindet, wird in der fünften Schicht das Gesamtergebnis durch Summenbildung erzeugt.

Zu beachten ist, daß in der vierten Schicht des Sogeno-Takagi-Kang-Reglers die Eingangswerte zur Berechnung der Funktionsergebnisse direkt mit einfließen. [Sau08]

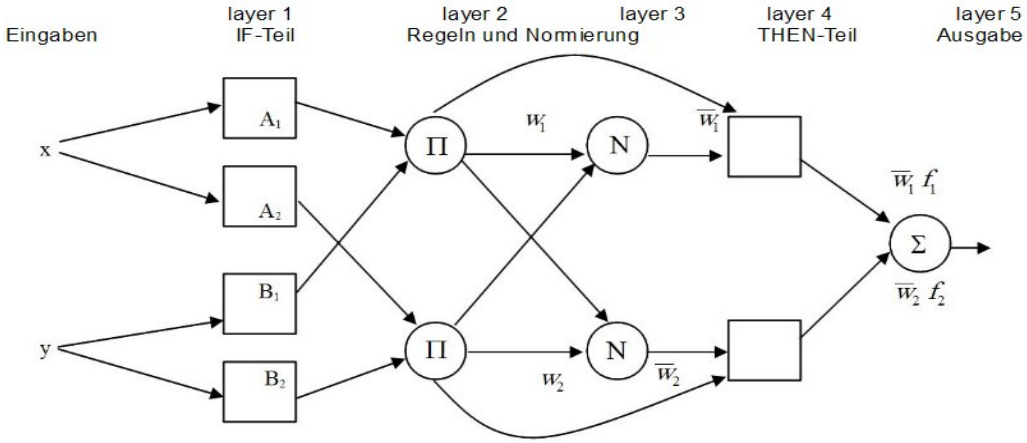

Abbildung 9: ANFIS-Architektur zur Nachbildung des Tsukamoto-Reglers [Sau08]

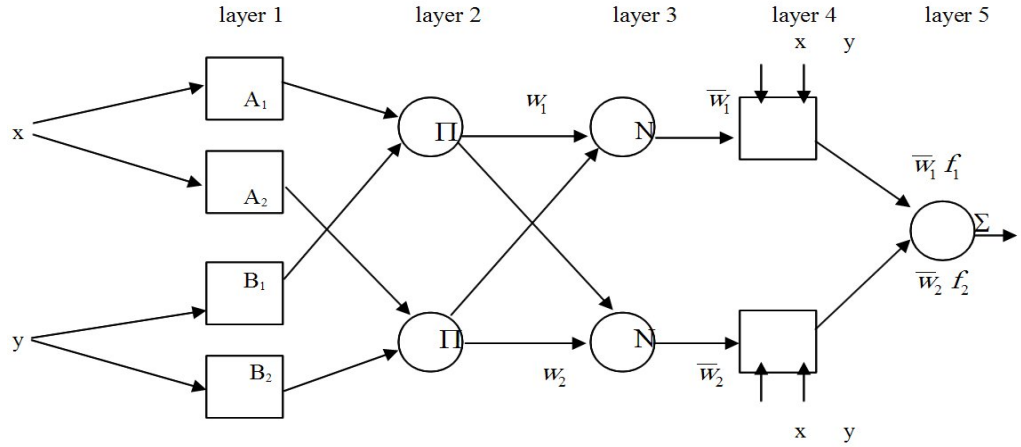

Abbildung 10: ANFIS-Architektur zur Nachbildung des Sugeno-Takagi-Kang-Regler [Sau08]

4.4 Hybrid-Lern-Algorithmus

ANFIS verwendet zur Optimierung seiner Parameter den sogenannten Hybrid-Lern-Algorithmus.
Dieser beinhaltet den Backpropagation-Algorithmus und die Methode des kleinsten quadratischen Schätzers. [Bot08]
Der Premissenparameter S_1, der ersten Schicht in den ANFIS-Modellen wird mittels des Backpropagationverfahren optimiert.
Der Konsequenzparameter S_2, der vierten Schicht in den ANFIS-Modellen wird mit der Methode des kleinsten quadratischen Schätzers optimiert. [Bor03]

Die Vorgehensweise ist dabei wie folgt:

Zur Optimierung des Konsequenzparameters S_2 wird ein Forward Pass durchgeführt.
Für jedes Trainingspaar (x, output) werden die Werte berechnet.
Der Premissenparameter S_1 bleibt unverändert, so daß die Funktion des Netzes nur vom Konsequenzparameter S_2 abhängig ist.
Diese Funktion ist bzgl. der Parameter bei festen Werten von S_1 linear.
Man erhält so zwei Matrizen A, B mit $A \cdot S = B$.
Nun wird der kleinste Quadratische Schätzer über die Minimierung von $\|AS - B\|^2$ berechnet.
Dies ergibt die optimierten Parameter von S_2. [Sau08]

Zur Optimierung des Premissenparameters S_1 wird ein Backward Pass durchgeführt.
Hierfür bleibt nun der Konsequenzparameter S_2 unverändert.
Zur Anwendung kommt nun der Backprobagation-Algorithmus als spezialfall der Gradientenmethode. [Sau08]

Bei ANFIS-Netzen, die den Tsukamoto-Regler nachbilden werden die Ereignisfunktionen der vierten Schicht durch stückweise lineare Funktionale ersetzt und dann jeweils die Hybrid-Methode angewandt.

Bei ANFIS Netzen, die den Sugeno-Takagi-Kang-Regler nachbilden, wird das Gesamtverhalten des Netzes als Linearkombination der Parameter der vierten Schicht interpretiert. [Mic02]

$$f = \frac{w_1}{w_1 + w_2} f_1 + \frac{w_2}{w_1 + w_2} f_2 = \overline{w}_1 f_1 + \overline{w}_2 f_2$$

5 Das NEFCON-Model

Das NEFCON-Model basiert auf dem dreischichtigen, generischen Fuzzy-Perzeptron. Dabei wird der Ansatz des reinforcement learnings[NNK99] verwendet. Das heißt, dass Regeln bzw. Fuzzy-Mengen je nach Güte belohnt oder bestraft werden.

Dabei kann das NEFCON-Model zwei Problemstellungen lösen. Zum einen kann es eine Regelbasis aufbauen. Vorraussetzung ist hierbei, dass die Fuzzy-Mengen korrekt und vollständig sind. Zum anderen kann es dazu verwendet werden Fuzzy-Mengen zu lernen, wobei eine vollständige Regelbasis vorrausgesetzt wird. Zusätzlich ist es nötig, dass die Anzahl der Fuzzy-Mengen bekannt ist.

5.1 Erlernen einer Regelbasis

Die Regelbasen, die optimiert werden sollen, lassen sich in drei Klassen unterteilen. Regelbasen die bisher leer sind, Regelbasen die vollständig sind im Sinne aller kombinatorischen Möglichkeiten und Regelbasen, die zufällig generiert wurden. Bisher ist allerdings nur das lernen duch leere und volle Regelbasen umgesetzt worden. Vgl. [NNK99]

Wird eine leere Regelbasis verwendet, werden zu Beginn alle kombinatorischen Möglichkeiten generiert und anschließend Regeln mit falschem Vorzeichen entfernt. Im Grunde erhält man dadurch also auch eine volle Regelbasis und somit lässt sich nun für beide möglichen Vorgehensweisen der gleiche Ablauf definieren.

Die Anschließende vorgehensweise lässt sich nun in zwei Phasen aufteilen. In Phase 1 wird eine feste Periode oder Anzahl an Iterationen definiert und dabei werden Regeln entfernt, wenn diese nicht zur optimalen Ausgabe passen.

In Phase 2 werden Regeln, die die gleiche Prämisse enthalten in Regelgruppen zusammengeführt um anschließend zufällig eine Regel der Regelgruppe auszuwählen. Jede dieser ausgewählten Regeln wird anahand ihrer Gewichtung ein Fehleranteil am Ergebnis berechnet. Anschließend werden alle Regeln, bis auf die mit dem kleinsten Fehleranzahl aus der Regelgruppe entfernt, wie auch Regeln die selten verwendet wurden.

5.1.1 Beispiel des erlernens einer Regelbasis

Um das Lernverfahren zu veranschaulichen wird ein Beispiel verwendet das aus [Lip06] entnommen ist. Dabei wird in diesem Beispiel ein inverses Pendel verwendet, dass sich auf einem Wagen bewegt. Ziel ist es nun geeignete Regeln zu finden, die das Pendel bei der Fahrt des Wagens ausbalanciert und es somit nicht umkippt.

Die durch dieses Lernverfahren entstandenen Regeln sind wiefolgt:

R_{01} : $IF\ x_1 = ng\ UND\ x_2 = ng\ THEN\ y = ng$

R_{02} : $IF\ x_1 = nm\ UND\ x_2 = ng\ THEN\ y = ng$

R_{03} : $IF\ x_1 = nk\ UND\ x_2 = ng\ THEN\ y = ng$

R_{04} : $IF\ x_1 = nm\ UND\ x_2 = nm\ THEN\ y = nk$

R_{05} : $IF\ x_1 = nk\ UND\ x_2 = nm\ THEN\ y = nk$

R_{06} : $IF\ x_1 = nn\ UND\ x_2 = nm\ THEN\ y = nm$

R_{07} : $IF\ x_1 = pn\ UND\ x_2 = nm\ THEN\ y = nk$

R_{08} : $IF\ x_1 = pk\ UND\ x_2 = nm\ THEN\ y = ng$

R_{09} : $IF\ x_1 = pm\ UND\ x_2 = nm\ THEN\ y = ng$

R_{10} : $IF\ x_1 = ng\ UND\ x_2 = nk\ THEN\ y = ng$

R_{11} : $IF\ x_1 = nm\ UND\ x_2 = nk\ THEN\ y = nm$

R_{12} : $IF\ x_1 = nk\ UND\ x_2 = nk\ THEN\ y = nn$

R_{13} : $IF\ x_1 = nn\ UND\ x_2 = nk\ THEN\ y = nk$

R_{14} : $IF\ x_1 = ng\ UND\ x_2 = nn\ THEN\ y = ng$

R_{15} : $IF\ x_1 = nm\ UND\ x_2 = nn\ THEN\ y = ng$

R_{16} : $IF\ x_1 = nk\ UND\ x_2 = nn\ THEN\ y = ng$

R_{17} : $IF\ x_1 = nn\ UND\ x_2 = nn\ THEN\ y = nn$

R_{18} : $IF\ x_1 = pn\ UND\ x_2 = pn\ THEN\ y = pn$

R_{19} : $IF\ x_1 = pk\ UND\ x_2 = pn\ THEN\ y = pm$

R_{20} : $IF\ x_1 = pm\ UND\ x_2 = pn\ THEN\ y = pk$

R_{21} : $IF\ x_1 = pk\ UND\ x_2 = pk\ THEN\ y = pn$

R_{22} : $IF\ x_1 = pk\ UND\ x_2 = pk\ THEN\ y = pk$

R_{23} : $IF\ x_1 = pm\ UND\ x_2 = pk\ THEN\ y = pg$

R_{24} : $IF\ x_1 = pn\ UND\ x_2 = pm\ THEN\ y = pg$

R_{25} : $IF\ x_1 = pk\ UND\ x_2 = pm\ THEN\ y = pk$

R_{26} : $IF\ x_1 = pm\ UND\ x_2 = pm\ THEN\ y = pg$

Das Erlernen der Regelbasis ist dabei vorallem davon abhängig, wie gut die Parameter gewählt wurden. Wichtig ist dabei die Wahl der Anzahl, der Iterationen durch Phase 1, sowie Phase 2. Empfohlen werden dabei 2000 bis 300 Iterationen. Außerdem sollte eine geeignete Stellgröße für die Minimalverwendung einer Regel gefunden werden. Dieser Wert sollte dabei zwischen 1.00 und 1.03 gewählt werden, da sonst wichtige Regeln, die aber nur selten vorkommen aus der Regelbasis entfernt werden. Zusätzlich ist es möglich bekannte Regeln zu definieren. Diese werden dann vom Lernalgorithmus nicht entfernt und es werden auch keine Regeln erzeugt, die die gleiche Prämisse enthalten.[Lip06]

5.2 Erlernen von Fuzzy-Mengen

Ein weiterer Einsatzbereich des NEFCON-Models ist das Erlernen von Fuzzy-Mengen. Dabei wird vorrausgesetzt, dass eine korrekte Regelbasis besteht. Durch den Fuzzy-Backpropagationalgorithmus wird die Güte einer Regel bestimmt um “gute” Regeln zu belohnen und “schlechte” Regeln zu bestrafen.

Dabei wird der Beitrag t_k der Regel R_k zum Ergenbis durch $t_k = u_k^{-1}(o_k)$ ermittelt. Die dadurch erhaltene Stellgröße gibt den Zustand der größe an wobei der Wert 0 dem optimalen Zustand entspricht. Enthält die Stellgröße dabei das richtige Vorzeiche, wird die Regel als gut eingestuft. Mit dieser Klassifizierung wird der Einfluß “guter” Regeln vergrößert und der Einfluß “schlechter” Regeln verkleinert.

Das Lernverfahren selbst besteht aus sechs Einzelschritten:

Schritt 1 Berechne Ausgabe o für die Meßwert, wende sie auf das zusteuernde System an und berechne die sich daraus ergebenden Meßwerte.

Schritt 2 Berechne den Fuzzy-Fehler, der sich aus den Meßwerten von Schritt 1 ergibt.

Schritt 3 Bestimme das Vorrücken des Stellwertes im neuen Systemzustand.

Schritt 4 Berechne für Regel R_k den Beitrag t_k. Dabei ist das Fehlersignal F_k für Regel R_k gegeben durch

$$F_k := \begin{cases} -o * E & \text{Vorzeichen von } t_k \text{ richtig} \\ o * E & \text{Vorzeichen von } t_k \text{ falsch} \end{cases}$$

Schritt 5 Modifizierung aller Eingabe-Fuzzy-Mengen $\tilde{A}_{i*}^{(k)} = (l_{i*}^{(k)}, m_{i*}^{(k)}, r_{i*}^{(k)})$.

$$\Delta l_{i*}^{(k)} = -\eta * F_k * (m_{i*}^{(k)} - l_{i*}^{(k)})$$
$$\Delta r_{i*}^{(k)} = -\eta * F_k * (r_{i*}^{(k)} - m_{i*}^{(k)})$$

Dabei stehen l, m und r für die linke, rechte und mittlere Fuzzy-Menge, die in diesem Beispiel verwendet wird. Diese Fuzzy-Mengen werden um den Δ verschoben um ein optimale Fuzzy-Menge zu erhalten. η entspricht der Gewichtung der Fuzzy-Menge. Allerdings wird Fuzzy-Menge m nicht neu berechnet, sondern als korrekt angesehen.

Schritt 6 Modifizierung aller Ausgabe-Fuzzy-Mengen $\tilde{B}_{i*}^{(k)} = (m_{i*}^{(k)}, b_{i*}^{(k)})$.

$$\Delta b_{j*}^{(k)} \begin{cases} \eta * F_k * (b_{j*}^{(k)} - m_{j*}^{(k)}) & b_{j*}^{(k)} > m_{j*}^{(k)} \\ \eta * F_k * (m_{j*}^{(k)} - b_{j*}^{(k)}) & b_{j*}^{(k)} < m_{j*}^{(k)} \end{cases}$$

5.3 Nachteile des NEFCON-Models

Ein großer Nachteil des NEFCON ist die Vorraussetzung von monotonen Fuzzy-Mengen. Das heißt das keine Gauß- & Dreiecks-Mengen möglich, diese werden von Fuzzy-Controllern allerding häufig verwendet. Ein weiterer Nachteil ist, dass es nur möglich ist ein Ausgabeneuron zu verwenden und es somit auch nur einen Ausgabewert gibt. Zusätzlich ist es nicht möglich die erzeugten Regeln zu überprüfen oder fehlende Fuzzy-Mengen zu generieren.[Lip06]

Literatur

[Bar05] BARNERT, Mrkus: Neuronalen Netze und das Fuzzy-Inferenz-System als Grundlagenerläuterung von Adaptive-Network-Based Fuzzy Inference Systemen. (2005), S. 15–18

[BKKN03] BORGELT, Christian ; KLAWONN, Frank ; KRUSE, Rudolf ; NAUCK, Detlef: *Neuro-Fuzzy-Systeme*. Wiesbaden : Vieweg Verlag, 2003

[Bot08] BOTHE, Hans-Heinrich: *Neuro- Fuzzy-Methoden: Einführung in Theorie und Anwendungen*. Springer Verlag, 2008. – 230–236 S.

[Kar10] KARTHIKEYAN, Dr. T.: Efficient Bio Metric IRIS Recognition System Using Fuzzy Neural Network. In: *Int. J. of Advanced Networking and Applications* 01 (2010), S. 371–376

[Lip06] LIPPE, Wolfram-Manfred: *Soft-Computing*. examen press, 2006

[Mic02] MICHELS, Kruse N. Klawonn: *Fuzzy-Regelung: Grundlagen, Entwurf, Analyse*. Springer, 2002. – 367–369 S.

[NNK99] NÜRNBERGER, A. ; NAUCK, D. ; KRUSE, R.: Neuro-fuzzy control based on the NEFCON-model: recent developments. In: *Soft-Computing* 2 (1999)

[Sau08] SAUER, Jürgen: Neuronale Netze, Fuzzy Control-Systeme und Genetische Algorithmen. (2008), S. 273–275