Prognosesysteme für die Verkehrssicherheit mit Methoden des Soft-Computings am Beispiel einer Glätteprognose und einer Fahrzustands-bestimmung

vom Fachbereich Elektrotechnik und Informatik
der Universität Siegen
zur Erlangung des akademischen Grades

DOKTOR-INGENIEUR (Dr.-Ing.)

Genehmigte Dissertation

von
Diplom-Ingenieur

Frank Wieland

geboren am 29.05.1969 in Lüdenscheid-Hellersen, Deutschland

I. Gutachter : Professor Dr.-Ing. Dr. h.c. K.W. Bonfig
II. Gutachter : Professor Dr.-Ing. H. Roth

Vorsitzender der Prüfungskommission : Professor Dr. rer. nat. R. Patsch

Tag der mündlichen Prüfung 18.12.2001

urn:nbn:de:hbz:467-77

Vorwort

Die vorliegende Dissertation entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Meßtechnik des Fachbereiches Elektrotechnik und Informatik der Universität Siegen.

Herrn Professor Dr.-Ing. Dr. h.c. K.W. Bonfig danke ich für die Betreuung und die Möglichkeit, diese Arbeit an seinem Institut durchzuführen. Ich bedanke mich besonders für seine freundliche und wohlwollende Unterstützung. Seine vielen Hinweise und Diskussionsbeiträge waren mir eine große Hilfe bei der Bearbeitung dieses Themas.
Mein Dank gilt auch meinen Kollegen am Institut für Meßtechnik des Fachbereiches Elektrotechnik und Informatik der Universität Siegen für die gute Zusammenarbeit. Des Weiteren gilt mein Dank all denjenigen, die zum Erfolg dieser Dissertation mit beigetragen haben, besonders Herrn Priv.-Doz. Dr.-Ing. F. Aliew, Herrn Dipl.-Ing. W. Steinmann und Herrn Dipl.-Ing. Farhad Berton , welche mir mit Ihrer Diskussionsbereitschaft und Korrekturlesung sehr geholfen haben. Ebenso gilt mein Dank der Vielzahl von Studien- und Diplomarbeitern, welche mir mit Ihren Arbeiten gute Beiträge zu dieser Arbeit geliefert haben.

Herrn Professor Dr.-Ing. H. Roth danke ich für die Übernahme des Co-Referats und der damit verbundenen Mühen. Bei Herrn Professor Dr. rer. nat. R. Patsch bedanke ich mich für die Übernahme des Vorsitzes der Prüfungskommission.

Abstract

In this work new prognosis systems for road safety are presented. The systems enable the entry and optimization of dynamic behavior in traffic consideration. To get such results an adaptation procedure was developed and tested by optimized weighting values of knowledge base of a fuzzy system.

Furthermore in this work a procedure was developed, which is able to extend and optimize the knowledge base of a fuzzy system by genetic algorithms. Thereby it is not necessary to know and describe all connections between the available values. Also the learning algorithm for neural fuzzy networks by using α-cuts was extended with a genetic algorithm. This enables a faster learning of the network and the determination of the optimal results.

It was shown that with the help of the described methods errors in measured values can be compensated very well. Furthermore it was shown that an intelligent system is also able to determine not directly measurable values and make with these values sufficiently good prognoses. This was shown in this work with the smoothness prognosis. The implementation of the system shows that the prognosis accuracy is 95%.

With the implemented system for the jam prognosis the prognosis accuracy is 75%. This value was increased in the course of the work by inclusion of the drivers behavior to 86%. For use of the measured value this is sufficient. An adaptation, which was implemented in the course of the work, did not bring significant improvement. This was also expected, since the adaptable values are taken off by the user profile mainly.

In systems, for different users more difficult optimization tasks have to be implemented, since continuing changing from user profiles prepares large problems. An optimized system created, can represent a perfectly insufficient profile for another user. The beginnings for detecting the driver behavior, selected here, are to be forecastable for the next reactions and are sufficiently exact. Although both procedures are very different, they produce approximately the same results. The resulting membership functions are alike up to small differences.

Finally, the determined procedures are not only limited to these applications.

Inhaltsverzeichnis:

Abbildungsverzeichnis

1 Aufgabenstellung

Am Beispiel verschiedener Prognosesysteme sollen Möglichkeiten untersucht werden, in wie weit Methoden des Softcomputings es ermöglichen, bei komplexen und auch bei nicht exakt mathematisch beschreibbaren Systemen die notwendigen Messwerte und -signale zu erfassen, bzw. zu ermitteln. Dies tritt vor allem bei abgeschlossenen oder gekapselten Systemen auf, da hier in vielen Fällen die Möglichkeit fehlt, Sensoren nachträglich einzubauen.

Ein weiteres Problem sind auch die Messwerte, welche mit keinen auf dem Markt befindlichen Sensoren gemessen werden können. Dies sind z.B. Messwerte über die Glättewahrscheinlichkeit einer Straße. Sie sind über Zusammenhänge beschreibbar, aber eine mathematische Beschreibung oder ähnliches fehlt. Um diese Werte zu bestimmen, sind Methoden notwendig, welche ein Ergebnis auch anhand von unsicheren Eingangswerten ermitteln können.

Weiter ist es erforderlich, dass diese Systeme sich an verschiedene Umgebungen anpassen müssen, ohne dass hierfür langwierige Anpassungen nötig sind. Es sind ebenfalls Adaptionsmethoden zu untersuchen, die diese Aufgabenstellung lösen. Für diesen Fall bietet sich eine Adaption mit Hilfe der genetischen Algorithmen an.

Insbesondere soll eine Methode für Fuzzy-Logik-Systeme, welche nicht voll beschrieben sind, untersucht werden. Hierbei ist es notwendig, dass fehlende Beschreibungen und Verknüpfungen automatisch der Regelbasis hinzugefügt werden. Dies wird mit Hilfe der genetischen Algorithmen realisiert.

Es wird außerdem eine Adaptionsmethode für Fuzzy-Systeme bearbeitet, welche die Gewichtungen der einzelnen Regeln in der Wissensbasis verwendet, um sie zu optimieren.

2 Einleitung und Stand der Technik

In dieser Arbeit werden die Möglichkeiten des Soft-Computings für die vorliegende, in Kapitel 1 beschriebene, Aufgabenstellung angewendet. Das Soft-Computing umfasst die Methoden der Fuzzy-Logik, der Neuronalen Netze, der Neuro-Fuzzy-Technik, die genetischen Algorithmen, die Chaos-Theorie und die realisierbaren Kombinationen dieser Methoden. Der Vorteil der Kombination mehrerer Methoden liegt darin, dass sich die Einzelvorteile ergänzen und Unzulänglichkeiten separat eingesetzter Systeme hierdurch kompensiert werden können.

Als erste praktische Anwendung wurde die Glätteprognose bearbeitet. Es existiert eine Vielzahl von realen Messwerten. Es soll aber für einen gewissen Zeitraum eine Prognose gemacht werden, welche auch die Unwägbarkeiten der Wetterkomponenten und Messsensoren und die nicht mathematisch beschreibbaren Zusammenhänge berücksichtigt. Hierfür ist ein intelligentes System notwendig. Frühere Versuche, der Firma Micks [28] mit dynamischen Modellen haben gezeigt, dass es sehr problematisch ist, solche Prognosen selbstoptimierend aufzubauen. Da solche Prognosesysteme in der Regel auf Autobahnen oder Flughäfen installiert werden, wo bei jedem Standort unterschiedliche Umgebungseinflüsse sind, wäre ein System, das sich anpassen und optimieren kann sehr von Vorteil.

Für diese Anwendung sind an verschiedenen Stellen Sensoren in den Untergrund bzw. die Fahrbahn eingelassen. Sollte sich die Messstelle auf einer Bergkuppe befinden, weil an dieser Stelle aufgrund starker Winde mit Eis zu rechnen ist, sind hier eine ganze Reihe von speziellen Einflüssen zu beachten. Diese Einflüsse stimmen aber nicht mit einer Messstelle überein, die in einer Talsenke zu finden ist. Hier ist der Grund eher der geringe Sonneneinfall, gerade in der Winterzeit. Das System soll aber nach Möglichkeit, diese Einflüsse erkennen und adaptieren.

Im Laufe der hier gemachten Untersuchungen zeigte sich, dass für solche Aufgabenstellungen ein nahezu einheitlicher Ansatz mit den Methoden des Soft-Computings möglich ist.

Als weiteres wird die Bestimmung der Stauwahrscheinlichkeit aufgrund fahrzeuginterner messbarer Parameter wie Geschwindigkeiten, Pedalbetätig-

ungsfrequenz und der Nachbildung des Fahrerverhaltens, usw. erläutert. Diese Werte können nur aufgrund umfangreicher Versuche ermittelt werden. Es ist daher notwendig, die Anzahl der Versuche und die Zeit, die benötigt wird, zu minimieren. Wenn diese Informationen ermittelt werden können, ist eine intelligente Motorsteuerung in der Lage vorausschauend auf die Verkehrssituation zu reagieren, den Benzinverbrauch und die Schadstoffabgabe zu minimieren, indem z.B. die Kraftstoffzufuhr genauer dosiert wird. Das ist das Ziel nahezu jeder modernen Motorsteuerung.

Hierfür ist es allerdings auch notwendig ein System zu entwerfen, welches ebenfalls aufgrund fahrzeuginterner messbarer Werte das Fahrerverhalten ermittelt. Auch hier können nur auf Grund umfangreicher Versuche solche Profile bestimmt werden. Da solche Verläufe nicht stetig sind, ist hier eine Optimierung schwieriger. Auch an dieser Stelle helfen die Methoden des Soft-Computings weiter. Das grundsätzliche Problem liegt darin, dass eine Vielzahl von unterschiedlichem Verhalten möglich ist. Ein Fahrer kann entweder auf Grund einer Notfallsituation hektisch reagieren, oder aus dem Grund, dass sein Fahrstil eher aggressiv ist. In diesem Fall ist es von Vorteil nicht die Kraftstoffeinspritzung auf Vollgas zuzulassen, da der Kraftstoff nur unnötig im nächsten Bremsvorgang verbraucht wird, ohne eine Leistungssteigerung zu erzielen. Ein optimales Drehmoment oder eine optimale Beschleunigung, in Bezug auf die Fahrgeschwindigkeit, bringt für alle Beteiligten einen größeren Vorteil.

Es existieren für die in dieser Arbeit behandelten Aufgaben verschiedene Ansätze. Hierzu zählen exakte, mathematische Modelle beziehungsweise dynamische Modellierungsansätze wie in [30-36] dargestellt. Insbesondere sind hier die Arbeiten [26] zu nennen. Auch vereinzelte Ansätze mit Fuzzy-Logik gibt es in diesen Bereichen. Es sind aber nahezu alles Einzellösungen, welche nur für den jeweiligen Aufgabentyp entwickelt wurden. Allgemeine Ansätze findet man in [26]. Hierbei handelt es sich in erster Linie um Ansätze zur Identifizierung dynamischer Systeme. Ebenso gibt es Ansätze, Aufgaben mit Hilfe der Estimationstheorie zu lösen. In [27] sind solche Ansätze ausführlich beschrieben und erforscht worden, was aber nicht das Ziel dieser Arbeit ist.

Auf dem Gebiet des Soft-Computings sind besonders die Arbeiten [37-45] zu nennen, die sich in erster Linie mit Modellbildungen befassen.

Für den Bereich der Glätteprognose existiert zurzeit nur ein dynamisches Modell erster Ordnung, welches die Temperatur prognostiziert, und ein einfaches fuzzy basiertes System zur Glätteprognose [28]. Hierbei erfolgt die einzige Optimierung mittels einer Adaption des dynamischen Modells ohne Verwendung zusätzlicher Messwerte.

Ähnliches gilt auch für die Fahrzustandsbestimmung. Auch hierbei sind zurzeit nur dynamische Modelle erster Ordnung auf dem kommerziellen Markt zu finden [29].

3 Allgemein verwendete Ansätze und Methoden

In dieser Arbeit werden Systeme und Vorgehensweisen beschrieben, die es ermöglichen, nicht direkt messbare Werte und Systemverhalten zu erfassen, nachzubilden und zu optimieren. Des Weiteren sind einige Systeme in der Lage auch Messwerte nachzubilden, die nicht direkt messbar, aber für eine effektive Weiterverarbeitung sehr nützlich und manchmal sogar notwendig sind.

Auf Grund der System- bzw. Wertestruktur ist diese Vielzahl von Methoden nötig, um die erforderlichen Informationen zu ermitteln und sie zu beurteilen. Außerdem sind diese Verfahren notwendig um die Merkmale sowie die Optimierung und die Vorgehensweisen zu bestimmen. Die jeweiligen Verfahren, einzeln verwendet, liefern hier nicht die gewünschten Ergebnisse.

Bild 3.1 zeigt das Grundschema des Soft-Computing. Es zeigt sehr anschaulich inwieweit die Fuzzy-Logik die eigentliche Grundtechnik des Soft-Computing ist. Die meisten Anwendungen des Soft-Computing verwenden die Fuzzy-Logik weil es gerade hierdurch möglich wird, linguistische Formulierungen zu verwenden. Bei einer Vielzahl von Anwendungen ist sie sehr vorteilhaft, weil die Wissensermittlung beziehungsweise die Prozessbeschreibung nicht mathematisch vollständig sein muss. Es besteht auch die Möglichkeit mit Hilfe von Befragungen und Erfahrungen Lösungen zu finden. In der Regel können erfahrene Ingenieure und Mitarbeiter Verknüpfungen und Aussagen über Messwerte und –grössen besser verbal beschreiben als mathematisch. Allein schon aus dem Grund, dass nicht alle Zusammenhänge unbedingt mathematisch zwingend sind. Es kommen auch sehr oft Abhängigkeiten zwischen Messwerten vor, die im Vorhinein nicht erkennbar, bekannt waren oder erwartet wurden.

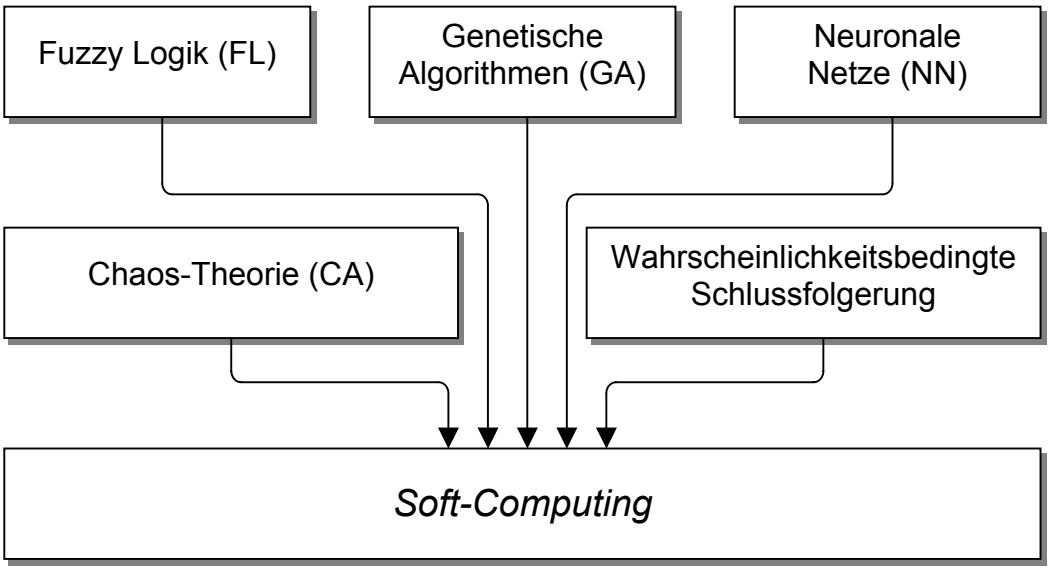


Bild 3.1: Grundschema des Soft-Computing

In den nachfolgenden Kapiteln werden nun die Grundtechniken soweit erklärt wie diese für die folgenden Arbeiten notwendig sind und verwendete Begriffe beschrieben. Es wird im Weiteren auf die verwendeten Grundeinstellungen und Grundjustierungen eingegangen.

Tabelle 3.1 [19] zeigt einen Vergleich zwischen den einzelnen Methoden des Soft-Computings.

	Fuzzy Sets	***Künstliche Neuronale Netze***	***Evolutionäres Computing, GA***
Schwächen	Wissensgewinnung Lernfähigkeit	Black Box Interpretations-möglichkeit	Codierung, Berechnungs-geschwindigkeit
Stärken	Interpretations-fähigkeit Transparenz Verständlichkeit Stufenweise Nachvollziehbarkeit Modellbildung Schlußfolgerungen Toleranzfähig Nachahmungen sind möglich	Lernfähigkeit Adaption Fehlertoleranz Kurvenanpassung Verallgemeinerung Approximationsfähig	Berechnungs-effektivität Globale Optimierung

Tabelle 3.1: Vergleich zwischen Komponenten des Soft Computings

3.1 Grundlagen der Fuzzy-Logik

Der Begriff „fuzzy“ stammt aus dem Englischen und bedeutet unscharf. Die verwendeten Methoden und Operatoren verhalten sich ähnlich wie gewöhnliche scharfe Mengen.

3.1.1 Grundlagen

Fuzzy-Mengen

Gegeben sei X als eine klassische Menge von Objekten, welche als Universalmenge bezeichnet wird. Die allgemeinen Elemente werden als x bezeichnet. Die Zugehörigkeit in einer klassischen Untermenge A von X wird oft als eine charakteristische Funktion μ_A von X betrachtet, so dass

$$\mu_A(x)=\begin{cases}1 \text{ für } x\in A\\ 0 \text{ für } x\notin A\end{cases} \tag{3.1}$$

Der Zugehörigkeitsgrad von x zu A ist $\mu_A(x)$. Je näher der Wert von μ_A an 1 liegt, desto mehr gehört x zu A [19].
Die Untermenge A wird komplett beschrieben durch die folgende Gleichung wobei für $x \in A$

$$A=\{(x, \mu_A(x)), x\in X\} . \tag{3.2}$$

Wenn X eine endliche Menge $\{x_1, \ldots , x_n\}$ ist, kann eine Fuzzy-Menge in X beschrieben werden mit

$$A=\mu_A(x_1)/x_1+\ldots+\mu_A(x_n)/x_n=\sum_{i=1}^{n} \mu_A (x_i)/x_i. \tag{3.3}$$

Zu der Gleichung 3.3 ist zu erwähnen, dass es sich hierbei nicht um eine Division handelt, sondern um einen Trennstrich zwischen den Werten, ebenso stellt die Summe die Vereinigung der Elemente dar.

Falls X infinit ist gilt

$$A=\int_{x} \mu_A(x)/x \,. \tag{3.4}$$

Falls $\forall\, x\in X\ \exists\, x^*$ mit der Bedingung

$$\mu(x^*) \geq \mu(x)$$

bzw.

$$\mu(x) \geq \mu(x^*)$$

hinreichend erfüllt ist, wird x^* als *maximaler* oder *minimaler* Wert der Zugehörigkeitsfunktion $\mu(x)$ bezeichnet. Es gilt [19]

$$\mu(x^*)=\max_{x\in X} \mu(x)$$

bzw. (3.5)

$$\mu(x^*)=\min_{x\in X} \mu(x)$$

Sollte der Punkt x^* nicht in X existieren, kann man die folgende Vereinbarung treffen: Es sind Folgen von x_1, x_2, ... in X zu finden, so dass

$$\lim_{i\to\infty} \mu(x_i) = \sup_{x\in X} \mu(x)$$

oder (3.6)

$$\lim_{i\to\infty} \mu(x_i) = \inf_{x\in X} \mu(x)$$

gilt.

Die Begriffe inf und sup bedeuten hier die größte untere bzw. die kleinste obere Grenze. Für zwei Fuzzy-Mengen A und B, die gleich sind, gilt

$$\forall\, x\in X,\ \mu_A(x)=\mu_B(x),\ A=B \tag{3.7}$$

Der *Träger* (Support) supp einer Fuzzy-Menge A ist die gewöhnliche Untermenge von X und wird beschrieben mit:

$$\text{supp } A=\{x \mid x\in X;\ \mu_A(x)>0\}. \tag{3.8}$$

Dies ist grafisch in Bild 3.2 [19] dargestellt:

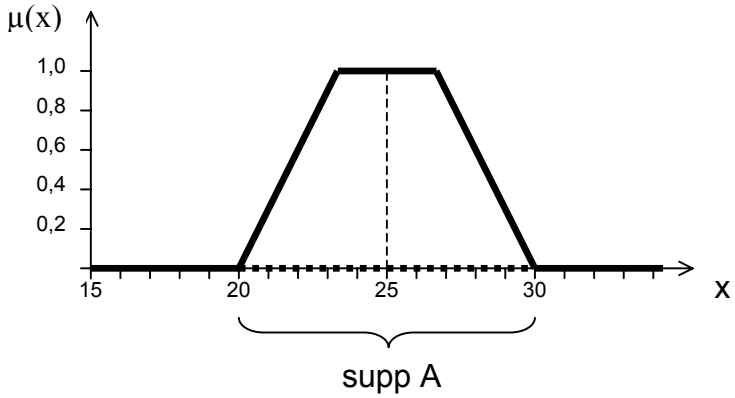


Bild 3.2: Träger einer Fuzzy-Menge A

Wenn eine Fuzzy-Menge aus einem einzelnen Punkt mit $\mu_A=1,0$ besteht, wird sie als *Singleton* bezeichnet.

In Bild 3.3 ist eine normalisierte Fuzzy-Menge dargestellt. Von einer normalisierten Fuzzy-Menge spricht man, wenn mindestens ein Wert die Zugehörigkeit von 1 hat.

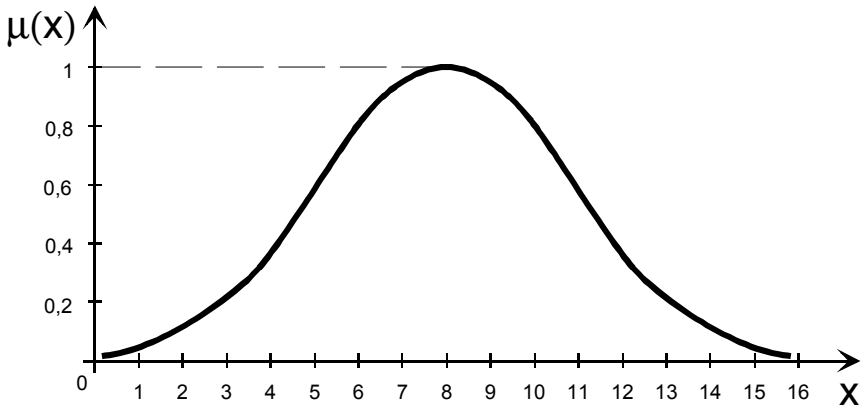


Bild 3.3: Normalisierte Fuzzy-Menge

In Bild 3.4 ist eine subnormale Fuzzy-Menge dargestellt. Von einer subnormalen Fuzzy-Menge spricht man, wenn kein Wert die Zugehörigkeit von 1 hat.

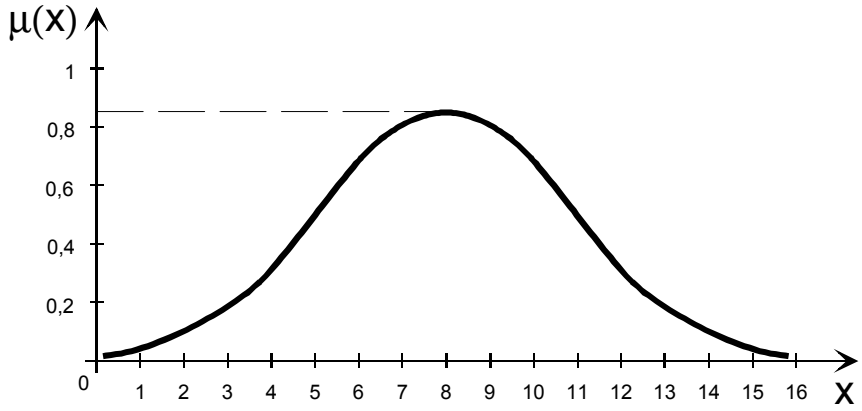


Bild 3.4: Subnormale Fuzzy-Menge

Die leere Menge $\varnothing$ wird definiert mit $\forall\, x \in X$, $\mu_{\varnothing}(x)=0$;

Definition: α-Schnitte von Fuzzy-Mengen

Die Grundmenge von Elementen, deren Zughörigkeitsfunktionswerte größer sind als eine Schwelle $\alpha \in (0,1]$, ist der α-Schnitt A^{α} von A:

$$A^{\alpha}=\{x \in X,\ \mu_A(x) \geq \alpha\} \tag{3.9}$$

Die *Strenge* des α-Schnittes $\overline{A}^{\alpha}$ wird definiert als

$$\overline{A}^{\alpha}=\{x \in X, \mu_A(x)>\alpha\} \tag{3.10}$$

Die α-Schnittmenge wird definiert durch

$$A^{\alpha}=\{(x,\ \mu_A{}^{\alpha}(x)=\mu_A(x)),\ x \in X,\ \alpha \in [0,1]\} \tag{3.11}$$

In diesem Fall gilt, dass μ_A eine Zugehörigkeitsfunktion einer Fuzzy-Menge A und $\mu_A{}^{\alpha}$ eine Zugehörigkeitsfunktion der α-Schnitt-Menge A^{α} ist. A^{α} ist eine α-Schnitt-Menge in der Form

$$A^{\alpha}=\{(x, \mu_A(x) \geq \alpha), x \in X\} \tag{3.12}$$

Ein weiterer wichtiger Punkt ist die Vereinigung von Fuzzy-Mengen. Hierfür ist es wichtig eine gewisse Anzahl von Grundoperationen ($\cup$, $\cap$) für Fuzzy-Mengen zu definieren [16].

1. Neutralelemente

$$\min(1,\mu_A(x))=\mu_A(x) \Rightarrow G \cap A=A$$

$$\max(0,\mu_A(x))=\mu_A(x) \Rightarrow \varnothing \cup A=A.$$

2. Kommutivität

$$\min(\mu_A(x),\mu_B(x))=\min(\mu_B(x),\mu_A(x)) \Rightarrow A \cap B=B \cap A$$

$$\max(\mu_A(x),\mu_B(x))=\max(\mu_B(x),\mu_A(x)) \Rightarrow A \cup B=B \cup A$$

3. Assoziativität

$$\min(\min(\mu_A(x),\mu_B(x)),\mu_C(x))=\min(\mu_A(x),\min(\mu_B(x),\mu_C(x)))$$
$$\Rightarrow (A \cap B) \cap C=A \cap (B \cap C)= A \cap B \cap C$$

$$\max(\max(\mu_A(x),\mu_B(x)),\mu_C(x))=\max(\mu_A(x),\max(\mu_B(x),\mu_C(x)))$$
$$\Rightarrow (A \cup B) \cup C=A \cup (B \cup C)$$

4. Monotonie

$$\mu_A(x) \leq \mu_C(x) \wedge \mu_B(x) \leq \mu_D(x)$$
$$\Rightarrow \min(\mu_A(x),\mu_B(x)) \leq \min(\mu_C(x),\mu_D(x))$$
$$\Rightarrow A \subset C \wedge B \subset D \Rightarrow A \cap B \subset C \cap D$$

$$\max(\mu_A(x),\mu_B(x)) \leq \max(\mu_C(x),\mu_D(x))$$
$$\Rightarrow A \subset C \wedge B \subset D \Rightarrow A \cup B \subset C \cup D$$

5. Idempotenz

$\min(\mu_A(x),\mu_A(x))=\mu_A(x) \Rightarrow A \cap A=A$

$\max(\mu_A(x),\mu_A(x))=\mu_A(x) \Rightarrow A \cup A=A$

6. Distributivität

$\min(\mu_A(x),\max(\mu_B(x),\mu_C(x)))=\max(\min(\mu_A(x),\mu_B(x),\min(\mu_A(x),\mu_C(x)))$
$\Rightarrow A \cap (B \cup C)=(A \cap B) \cup (A \cap C)$

$\max(\mu_A(x),\min(\mu_B(x),\mu_C(x)))=\min(\max(\mu_A(x),\mu_B(x)),\max(\mu_A(x),\mu_C(x)))$
$\Rightarrow A \cup (B \cap C)=(A \cup B) \cap (A \cup C)$

7. Absorption

$\min(\mu_A(x),\max(\mu_A(x),\mu_B(x)))=\mu_A(x) \Rightarrow A \cap (A \cup B)=A$

$\max(\mu_A(x),\min(\mu_A(x),\mu_B(x)))=\mu_A(x) \Rightarrow A \cup (A \cap B)=A$

8. De Morgansche Gesetze

$1-\min(\mu_A(x),\mu_B(x))=\max(1-\mu_A(x), 1-\mu_B(x)) \Rightarrow \overline{A \cap B} = \overline{A} \cup \overline{B}$

$1-\max(\mu_A(x),\mu_B(x))=\min(1-\mu_A(x), 1-\mu_B(x)) \Rightarrow \overline{A \cup B} = \overline{A} \cap \overline{B}$

9. Doppeltes Komplement

$1-(1-\mu_A(x)) = \mu_A(x) \Rightarrow \overline{\overline{A}} = A$

10.Komplement von Basis und leerer Menge

$1-1=0 \Rightarrow \overline{G} = \varnothing$
$1-0=1 \Rightarrow \overline{\varnothing} = G$

Es gilt weiter: A und B sind zwei Fuzzy-Mengen in X, mit den Zugehörigkeitsfunktionen μ_A und μ_B. Die Mengenlehren-Operationen Vereinigung, Schnittmenge und Komplement für Fuzzy-Mengen sind über ihre Zugehörigkeitsfunktionen definiert.

Definition: Vereinigung

Eine Vereinigung ($\cup$) der Fuzzy-Mengen A und B wird wie folgt berechnet:

$$\forall\, x \in X,\ \mu_{A\cup B}(x) = \max(\mu_A(x),\mu_B(x)) \tag{3.13}$$

hierbei ist $\mu_{A\cup B}$ die Zugehörigkeitsfunktion von $A \cup B$.

Definition: Schnittmenge

Die Zugehörigkeitsfunktion $\mu_{A\cap B}$ wird beschrieben mit:

$$\forall\, x \in X,\ \mu_{A\cap B}(x) = \min(\mu_A(x), \mu_B(x)) \tag{3.14}$$

Diese ist in Bild 3.5 dargestellt.

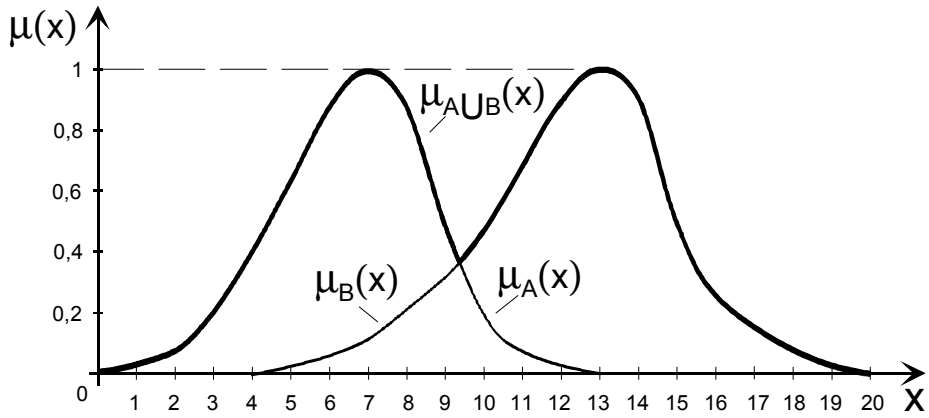


Bild 3.5: Vereinigung der Fuzzy-Mengen

Die Elementaren Operatoren, die in 3.13 und 3.14 beschrieben werden, wurden zuerst von Zadeh eingeführt.

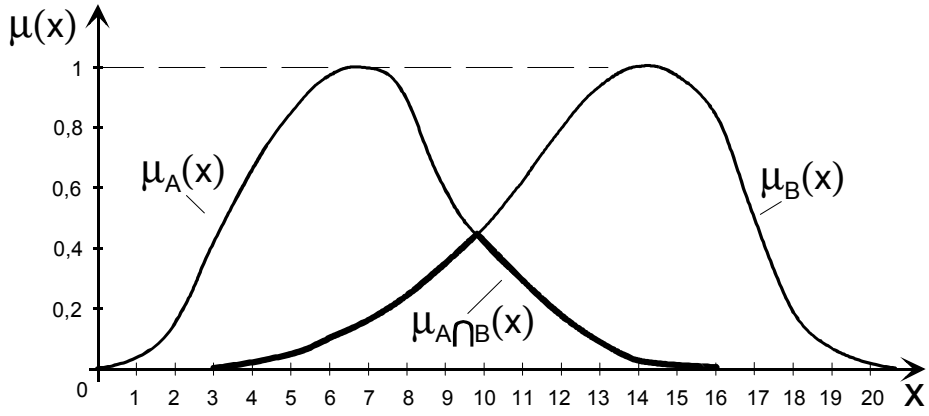


Bild 3.6: Schnittmenge der Fuzzy-Mengen

Für die Vereinigung und für Schnittmengen von Fuzzy-Mengen können auch andere Operatoren, wie wahrscheinlichkeitsabhängige Operatoren, Anwendung finden [19]. Die Formeln für die Produkt- und Wahrscheinlichkeitssumme für die Vereinigung und den Schnittpunkt lauten wie folgt.

Schnittmenge (Produkt)

$$\forall\, x \in X,\ \mu_{A\cdot B}(x) = \mu_A(x)\cdot\mu_B(x) \tag{3.15}$$

Man zählt diesen Operator zu der T-Norm [54].

Vereinigung (Wahrscheinlichkeitssumme)

$$\forall\, x \in X,\ \mu_{A\hat{+}B}(x) = \mu_A(x)+\mu_B(x)-\mu_A(x)\cdot\mu_B(x) \tag{3.16}$$

Man zählt diesen Operator zu der T-Conorm. Diese wird auch als S-Norm bezeichnet [54].

Definition: Komplement

Das Komplement $\overline{A}$ von A wird mit der folgende Zugehörigkeitsfunktion beschrieben:

$$\forall\, x \in X,\ \mu_{\overline{A}}(x) = 1-\mu_A(x) \tag{3.17}$$

Dieses ist in Bild 3.7 dargestellt.

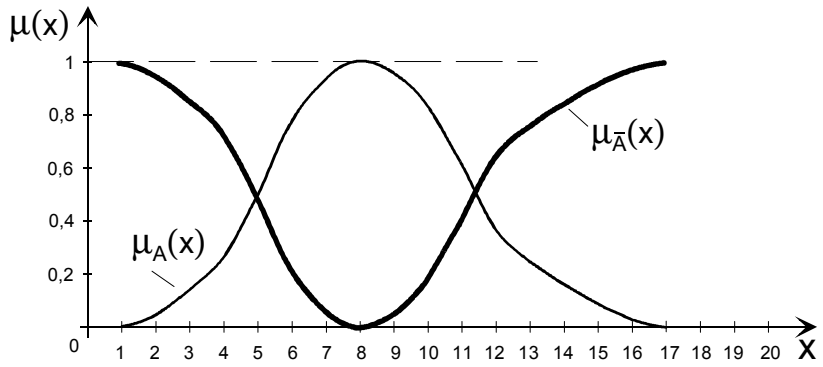


Bild 3.7: Komplement der Fuzzy-Mengen

Definition: Konzentration und Abschwächung (Dilatation) [19]

Es sei A eine Fuzzy-Menge der Grundmenge G gegeben:

$$A = \{(x,\mu_A(x))|x\in X\} \tag{3.18}$$

Durch Potenzieren mit dem Operator kon_n

$$kon_n A = \{(x;[\mu_A(x)]^n)|\ x\in X\} \tag{3.19}$$

erreicht man eine Konzentration von A. Dies ist in Bild 3.8 dargestellt.

Durch Potenzieren mit einem Bruch (Wurzelziehen) mit dem Operator dil_n

$$dil_n\ A = \{(x;\sqrt[n]{\mu_A(x)})|\ x\in X\} \tag{3.20}$$

erreicht man eine Abschwächung (Dilatation) von A. Dieses ist in Bild 3.9 dargestellt.

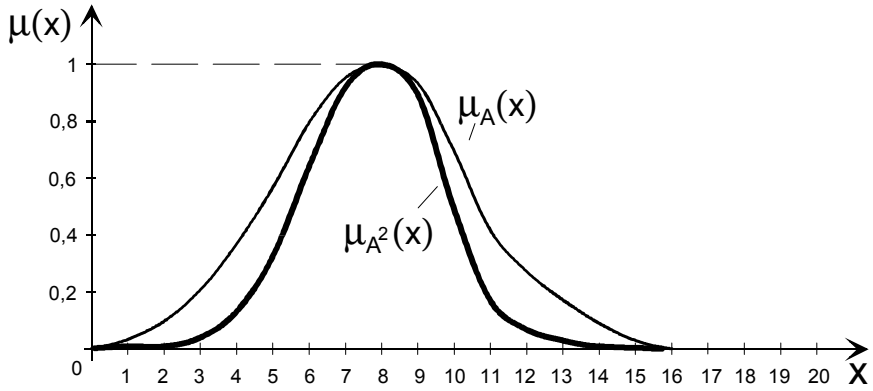


Bild 3.8: Konzentration der Fuzzy-Menge

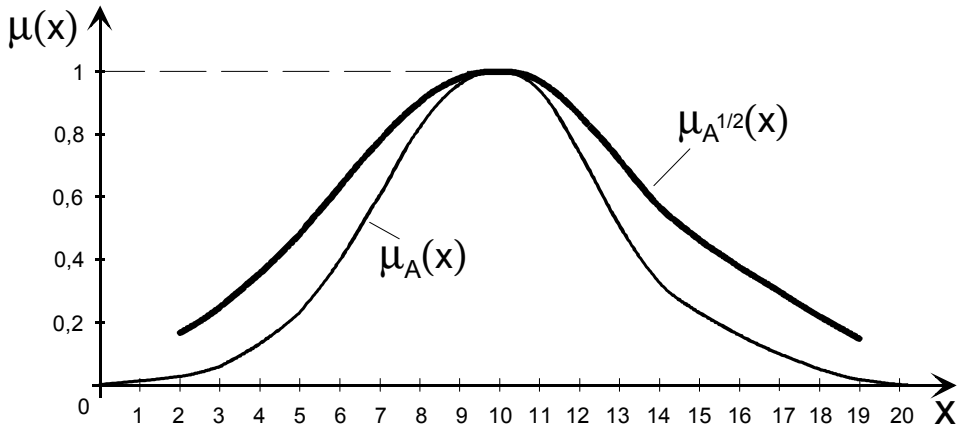


Bild 3.9: Abschwächung (Dilatation) der Fuzzy-Mengen

Nachfolgend sind einige Beschreibungen zusammengefasst, die den Begriff A sehr gut interpretieren [19]

A	= $\int \mu_A(x) / x$
sehr A	= $\int [\mu_A(x)]^2 / x$
sehr sehr A (außergewöhnlich A)	= $\int [\mu_A(x)]^4 / x$
etwas A (ziemlich A, mehr oder weniger A)	= $\int \sqrt{\mu_A(x)} / x$
ein bisschen A	= $\int \sqrt[4]{\mu_A(x)} / x$
nicht A	= $\int 1-\mu_A(x) / x$
nicht sehr A	= $\int 1-[\mu_A(x)]^2 / x$

3.1.2 Fuzzy-Relationen

Ein weiterer Bereich, der in dieser Arbeit Verwendung findet, sind die Fuzzy-Relationen, insbesondere die IF-Then Relationen.

IF-Then Fuzzy-Relation

Es seien A und B Fuzzy-Untermengen der gewählten Universen X und Y.

Um die Fuzzy-Untermengen A und B der ungleich gewählten Universen X und Y zu erhalten, wird das Konzept einer bedingten Fuzzy-Aussage eingeführt, so dass gilt:

$A \Rightarrow B$ oder „IF A THEN B“

Es wird dann die implizierte Relation R mit den Termen des kartesischen Produktes der Untermengen A und B ausgedrückt. Es gilt hier $R=A \times B$. Die Zugehörigkeitsfunktion ist definiert durch

$$\mu_R(x, y)=\mu_{A\times B}(x,y)=\min[\mu_A(x),\mu_B(y)],\ x\in X,\ y\in Y. \tag{3.21}$$

Dies wird auch als „AND“- Operator bezeichnet
Es ist möglich, die Fuzzy-Menge zu bestimmen. In solchen Fällen wird eine bedingte Fuzzy-Aussage festgelegt, die von der Form

IF A THEN IF B THEN C

ist.
Dann wird die Fuzzy-Relation R geschrieben als

$$R=A\times(B\times C)=A\times B\times C. \tag{3.22}$$

Fuzzy-Relationen können aus zwei Fuzzy-Bedeutungen erzeugt werden. Diese einfachen Bedeutungen werden durch die Benutzung der Verknüpfungsglieder „OR“, („ELSE“), „AND“, usw. verbunden.

3.1.3 Fuzzy-Funktion

Es sei eine scharfe (crispe) Funktion $f:R\rightarrow R$ gegeben. Dann wählt man ihre Graphen [19]

$$\{(x,y)\in R^2 \mid y=f(x)\} \tag{3.23}$$

als den Kern einer Fuzzy-Menge F, wobei deren Zugehörigkeitswerte im gleichmäßigen Abstand oberhalb und unterhalb des Graphen der Funktion liegen. Diese Fuzzy-Menge F stellt eine Fuzzy-Funktion dar. Für explizite Funktionen f können wir F als eine Familie (mit dem Familien Parameter x) von Fuzzy-Nummern Y(x) interpretieren. Jede besitzt {f(x)} als Kern und

$$\mu_{Y(x)}(y)=\mu_F(x,y) \tag{3.24}$$

als Zugehörigkeitsfunktion. Für den Fall einer implizierten Funktion f mit dem Graphen

$$\{(x,y)\in R \mid f(x,y)=0\} \tag{3.25}$$

könnte man regelmäßig eine mögliche klare Interpretation als eine scharfe (crispe) Grenze eines Gebietes benutzen, wenn man zu Fuzzy-implizierten-Funktionen übergeht. Diese Grenze ist dann sozusagen mit einem Graustufen-Band ausgestattet.

Es gibt drei Arten von Fuzzy-Funktionen:

– Normale Funktionen mit Fuzzy-Merkmalen oder hinreichenden Fuzzy-Einschränkungen;
– Funktionen, die nur die Unschärfe ihrer Argumente „tragen“, ohne sich selbst extra Unschärfe zu generieren: Das Bild einer Nicht-Fuzzy-Funktion ist eine Nicht-Fuzzy-Funktion;
– Weniger bekannte Funktionen von Nicht-Fuzzy-Argumenten: Das Bild eines Elementes ist unscharf durch das Verwackeln der Funktion.

Nun betrachten wir die Fuzzy-Funktion einer Nonfuzzy-Variablen. Hierbei können zwei Arten einer Fuzzy-Funktion von einer Nonfuzzy-Variablen betrachtet werden:

a) fuzzyfizierende Funktion

b) Fuzzy-Gruppe von Funktionen

Eine fuzzyfizierende Funktion von V nach W ist eine normale Funktion von V nach W der Art:

$P(W), \quad \tilde{f} : x \rightarrow \tilde{f}(x).$ (3.26)

Eine Fuzzy-Gruppe F der Funktionen von V nach W ist eine Fuzzy-Menge in W^V, die bei jeder Funktion f von V nach W einen Zugehörigkeitswert $\mu_F(f)$ in F hat. Es sei f eine Funktion von P(V) nach P(W). Ein Beispiel einer solchen Funktion f ist die Ausdehnung einer normalen Funktion von X nach Y.

3.1.4 Inferenzen

Die Schlußfolgerungen, welche aus den Regeln der Wissensbasis ermittelt werden, werden als Inferenz bezeichnet.

Die wohl am meisten verwendete Inferenz-Methode ist die Inferenz von Mamdani.

Die Mamdani-Inferenz wird auch als Maximum-Minimum-Inferenz bezeichnet. Hierbei werden die Terme der Ausgangsvariablen auf den Erfüllungsgrad der Vorbedingung begrenzt (Minimum). Die so erzeugten unscharfen Mengen werden dann zu einer einzigen zusammengefasst (Maximum).

Eine weitere Inferenz-Methode ist die von Larsen.

Sie wird auch als Maximum-Produkt Inferenz bezeichnet. Hierbei werden die Terme mit dem Erfüllungsgrad der Vorbedingung multipliziert (Produkt) und dann zusammengefasst (Maximum).

Beide Methoden unterscheiden sich im Ergebnis nach der Defuzzyfizierung nur gering.

3.1.5 Defuzzifizierung

Die Defuzzifizierung ist die Umwandlung eines Fuzzywertes wieder in eine scharfe (crispe) Zahl. Die Fuzzy Zahl liegt in der Regel in einer Zugehörigkeitsfunktion vor, welche hinsichtlich ihrer Lage und Form diese am besten charakterisiert. Dies kann mit einer Vielzahl von Methoden erfolgen [46]. Die wichtigsten Methoden sind:

- Maximum- bzw. Mustererkennungs-Methode: Die Vorschrift für diese Defuzzyfizierung besteht darin, nur diejenige Regel mit dem höchsten Erfüllungsgrad zu betrachten. Danach liefert die Auswertung dieser Regel, einen festen Werte als Ausgangsgröße. Dies bedeutet bei einem eindeutigen Maximum der Funktion, wie, z.B. bei Dreiecksfunktionen, ist das eindeutige Maximum der Ausgangswert. In den anderen Fällen bildet man den Mittelwert der existierenden Maxima. Bei einer Trapezfunktion wäre es der arithmetische Mittelwert der oberen Trapezeckpunkte.

- Schwerpunktmethode („Center of Gravity"- Methode): Der Flächenschwerpunkt der nach dem Inferenzschema resultierenden Ausgangs-Fuzzy-Menge wird über die Ausgangsgröße gebildet. Der Abszissenwert bildet dann die scharfe Ausgangsgröße.

  Die Schwerpunktmethode hat einen kleinen, aber nicht unbedingt zu vernachlässigenden Fehler. Da der Schwerpunkt der Fläche unter einer Zugehörigkeitsfunktion nie an deren Rändern liegen kann, es sei denn bei einem Dreieck, wird der Wertebereich der Ausgangsgröße nicht ganz ausgeschöpft. Das bedeutet es gibt Teile der Ergebnis-Fuzzy-Menge die unberücksichtigt bleiben (Bild 3.10).

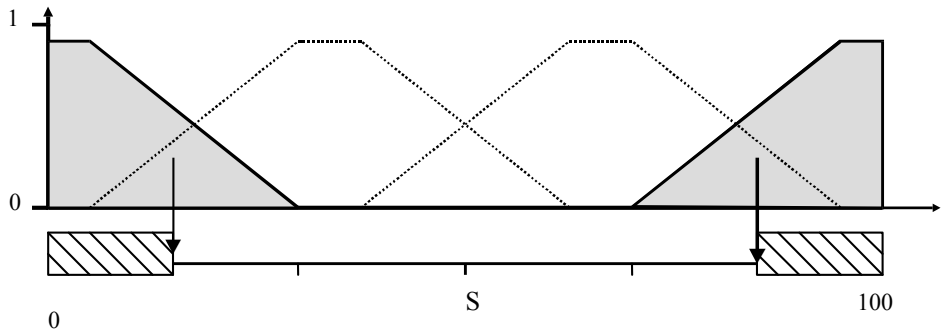


Bild 3.10: Schwerpunktmethode (die schraffierten Rechtecke kennzeichnen den nicht verwendbaren Bereich).

Das Problem hierbei ist, dass z.B. bei Motorsteuerungen nie der Maximalwert wie Vollgas oder Stillstand erreicht werden kann ohne zusätzliche Maßnahmen zu verwenden. Eine Variante wird nun nachfolgend beschrieben.

- Erweiterte Schwerpunktmethode: Die erweiterte Schwerpunktmethode schafft Abhilfe durch eine einfache fiktive, nur für die Schwerpunktberechnung

relevante symmetrische Erweiterung der an den Bereichsgrenzen gelegenen Zugehörigkeitsfunktionen. Durch eine Spiegelung der äußeren Fuzzy-Terme, ist der gesamte Wertebereich der Zugehörigkeitsfunktion für die Schwerpunktberechnung nutzbar (Bild 3.11).

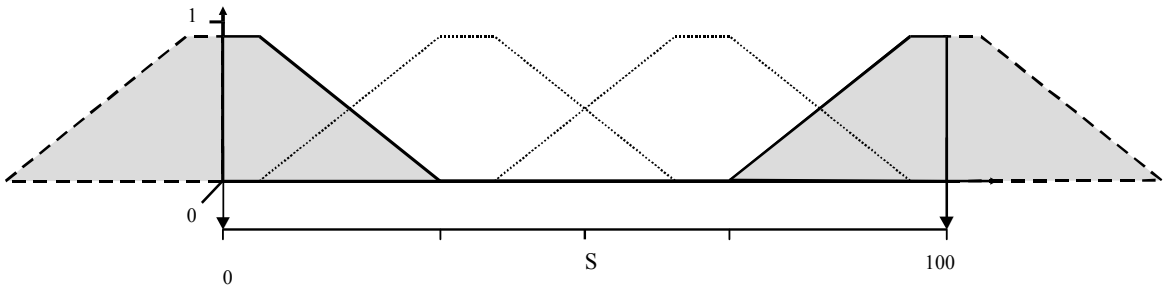


Bild 3.11: Erweiterte Schwerpunktmethode — voller Wertebereich

Beim Einsatz einer Schwerpunktmethode ist zu bedenken, dass eine Vielzahl numerischer Integrale für die Ermittlung des Flächenschwerpunktes berechnet werden müssen und dies unter Umständen sehr aufwendig ist, einen erhöhten Speicherplatzbedarf erfordert und in Echtzeitanwendungen zu Zeitproblemen führen kann. Eine kürzere Variante, die häufig als Näherung genommen wird, ist folgende:

- Schwerpunktmethode für Singletons: Diese Defuzzifizierungsmethode kann nur angewendet werden, wenn die Terme auf der Ausgangsgröße Singletons sind. Die scharfe Ausgangsgröße erhält man, indem die Produkte aus Erfüllungsgrad einer Regel und der Größe des Singletons über alle Regeln aufsummiert und durch die Summe der Erfüllungsgrade dividiert werden. Diese Defuzzyfizierungsmethode stellt einen Entartungsfall der Schwerpunktberechnung dar, der dort als Näherungsformel benutzt wird [18]. Sie bildet jedoch eine schnelle und Speicherplatz schonende Methode zur Ermittlung eines reellen Zahlenwertes (Bild 3.12).

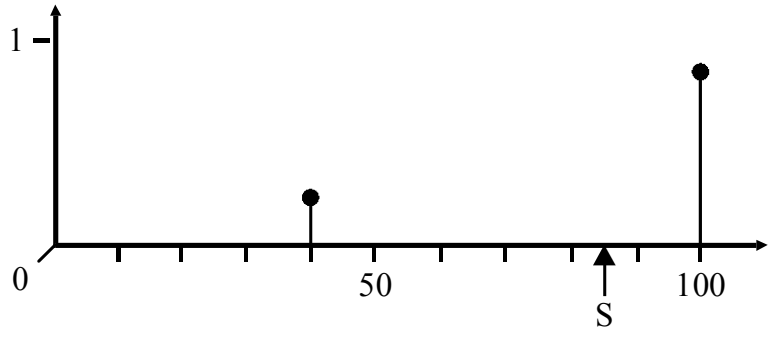


Bild 3.12: Schwerpunktmethode für Singletons

Die allgemeine Berechnungsvorschrift für die Koordinate x_S des Schwerpunktes $S(x_S/y_S)$ des unter der Funktion y = f(x) zwischen x = a und x = b gelegenen Flächenstücks ist aus der Mathematik bekannt [51].

$$x_s = \frac{\int_a^b x \cdot f(x)dx}{\int_a^b f(x)dx} \tag{3.27}$$

Daraus ergibt sich die Schwerpunktbestimmung für die scharfe Ausgangsgröße:

$$y_{res} = \frac{\int_0^\infty y \cdot \mu(y)_{res}\, dy}{\int_0^\infty \mu(y)_{res}\, dy} \tag{3.28}$$

Dies ist in Bild 3.13 dargestellt.

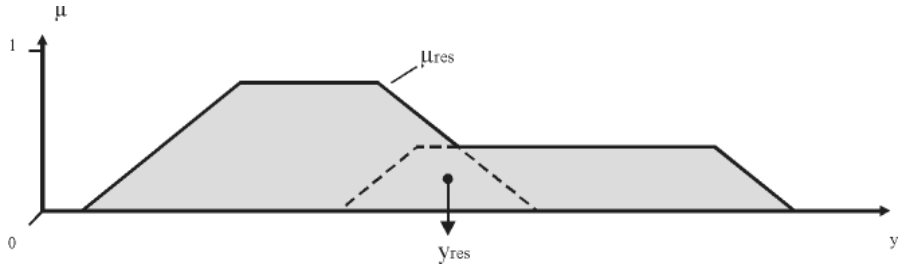


Bild 3.13: Defuzzifizierung nach der Schwerpunktmethode

Durch eine ausreichende Näherung werden die Abszissen y_i, der Schwerpunkte der Ausgangsmengen aller m Regeln, die dreiecks- oder trapezförmig sein sollten, in eine mit dem Erfüllungsgrad H_i gewichtete Summe eingebracht [46]. Dies ist in Bild 3.14 dargestellt.

$$y_{res} = \frac{\sum_{i=1}^{m} y_i H_i}{\sum_{i=1}^{m} H_i} \tag{3.29}$$

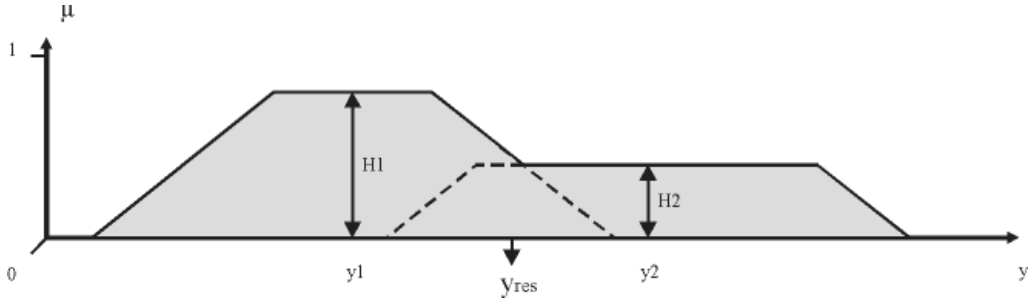


Bild 3.14: Anwendung der Näherungsformel für die Schwerpunktberechnung

3.2 Grundlagen der Theorie der Neuronalen Netze

Bei einigen Aufgabenstellungen ist es von großem Vorteil wenn das System in der Lage ist Zusammenhänge zu erkennen bzw. sie zur ermitteln. Es besteht dann unter Umständen die Möglichkeit Ereignisse und Einstellungen vorherzusehen oder zumindest zu erwarten. Ebenso kann hierdurch eine Voreinstellung vereinfacht werden.

Eine Möglichkeit hierfür ist die Theorie der Neuronale Netze. Ein Neuronales Netz verhält sich fast wie ein menschliches Gehirn. Aufgrund bestimmter Einflüsse und Anregungen werden entsprechende Ausgangssignale generiert. Um die gewünschten Ausgangssignale zu erreichen muss ein Netz definiert werden. Hierfür steht eine Vielzahl verschiedener Algorithmen zur Verfügung die sich in mehrere Gruppen unterteilen lassen. Als Beispiel ist die Gradientenmethode zu nennen.

3.2.1 Das menschliche Neuron

An dieser Stelle wird auf die Beschreibung theoretischer oder experimenteller Neuronen Modelle verzichtet da Sie für diese Arbeit nicht relevant sind. Ein einfaches menschliches Neuron ist in Bild 3.15 dargestellt [19].

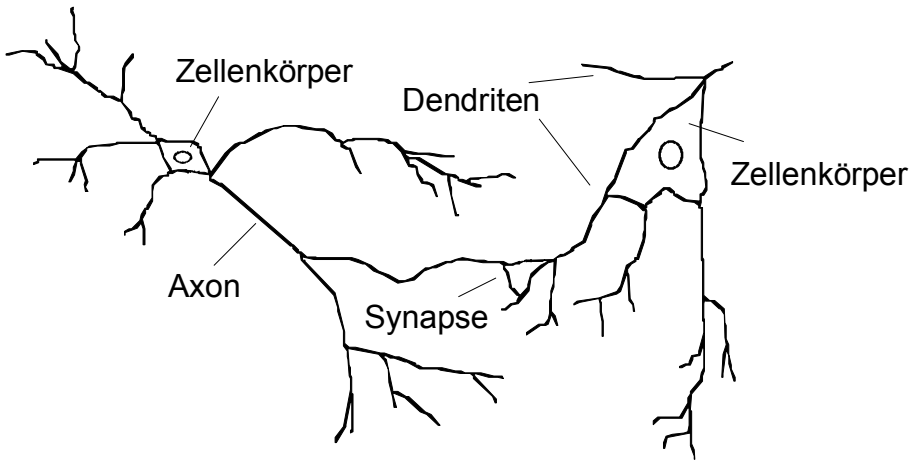


Bild 3.15: Menschliches Neuron

Die ersten Ansätze in diesem Bereich begannen 1947 kurz nachdem die ersten Arbeiten über die biologischen Grundzüge im menschlichen Gehirn veröffentlicht worden waren. Wirkliche Fortschritte gab es eigentlich erst als Rosenblatt im Jahre 1961 das Percepton-Model entwickelte. Hierauf aufbauend wurde eine Vielzahl von Arbeiten veröffentlicht. An dieser Stelle sind die Arbeiten von Grossberg, Kohnen und insbesondere Hopfield zu erwähnen [3, 19, 48, 49, 50].

Die wichtigste und herausragendste Eigenschaft der künstlichen Neuronalen Netze ist die des Lernens. Es sind aus diesem Grund eine Vielzahl von Anwendungen gerade in der Muster-, Bild- und Spracherkennung entstanden. Das Lernen bedeutet in den meisten Fällen trainieren. Dies wird mit Hilfe von Lernmustern realisiert, welche durch Ein- und Ausgangsmusterpaare bestimmt werden. Ist diese Lernmenge nur hinreichend groß genug, lassen sich hierdurch eine Vielzahl von Informationen gewinnen. Dieser Abschnitt wird Lern- oder Trainingsphase genannt. Eines der Probleme bei Neuronalen Netzen ist die Lernphase, da sie entweder sehr langwierig sein kann oder nicht genau genug gemacht werden kann, weil nicht genügend Trainingspaare zur Verfügung stehen.

Eine weitere Option ist die Möglichkeit der Adaption. Die während des Betriebs ermittelten Werte können wiederum für ein erneutes Training verwendet werden.

Solche Netze haben zusätzlich die Fähigkeit fehlerhafte und unvollständige Informationen zu verarbeiten. Es erfolgt eine Vervollständigung bzw. eine Korrektur von unvollständigen bzw. verrauschten Eingabemustern. Somit ist es möglich, verschwommene und unvollständige Informationen und Werte sinnvoll zu verarbeiten. So kann man bei dieser Eigenschaft eine enge Analogie zur menschlichen Informationsverarbeitung feststellen, was eine Überlegenheit gegenüber rein regelbasierten Systemen darstellt.

Aufgrund des Aufbaus neuronaler Netze stellen solche Netze eine massive Parallelität dar, d.h. die Neuronen können unabhängig voneinander arbeiten. Es bedeutet zusätzlich eine automatische Parallelisierung von Problemen auf Grund des Aufbaus.

Ein weiterer Aspekt ist die „Fehlertoleranz“ solcher Netze. Auf Grund einer Erhöhung der Anzahl der Neuronen hat man eine gewisse Redundanz in solchen Systemen.

3.2.2 Architektur und Aufbau neuronaler Netze

In Bild 3.16 ist das Neuron mathematisch dargestellt [19]. Beim Modellieren gewöhnlicher neuronaler Netze, bestehen diese aus einer großen Anzahl verbundener Neuronen. An die Neuronen wird der Eingangsvektor X der Eingangssignale x_1, x_2,..., x_n angelegt. Dieser Eingangsvektor entspricht den Signalen, die an den Synapsen biologischer Neuronen anliegen. Jedes Eingangssignal wird mit dem dazugehörigen Verbindungsgewicht w_1, w_2,..., w_n multipliziert. Die ist das Analogon der Synapsenwirksamkeit bei biologischen Systemen. Das Verbindungsgewicht ist ein Skalar: für reizende Verbindungen positiv und für hemmende negativ. Die mit den Verbindungsgewichten gewichteten Eingangssignale gelangen zu einem Summierer. Im Summierer werden die Eingangssignale algebraisch addiert und der Reizungsgrad des Neurons festgestellt. Dies geschieht mit der folgenden Formel

$$I = \sum_{i=1}^{n} w_i x_i \tag{3.30}$$

Das Ausgangssignal des Neurons wird über die folgende Funktion bestimmt:

$$y = F(I-\theta) \tag{3.31}$$

hierbei ist θ der Schwellenwert des Neurons.

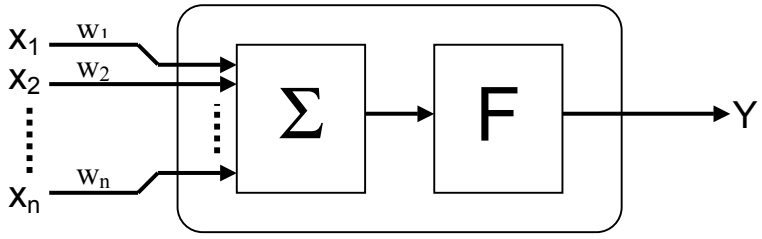


Bild 3.16: Mathematisches Modell des Neurons

Das einzelne Neuron entfaltet seine Möglichkeiten erst richtig, wenn es zusammen mit einer Vielzahl gleich aufgebauter Neuronen zu einem Netz zusammengeschaltet wird. Hierbei wird dann der Eingangsvektor durch die Aktivierung sogenannter Eingangsneuronen erzeugt.

Die Menge der Ausgangssignale der zusammenarbeitenden Neuronen ergeben den Ausgangsvektor Y. Die einzelnen Verbindungsgewichte w_{ij} bilden zusammen die Gewichtungsmatrix W. Hierbei ist anzumerken, dass w_{ij} das Verbindungsgewicht zwischen dem i-ten und j-ten Neuron des Netzes ist.

Es gibt eine Vielzahl verscheidender Netzstrukturen. An dieser Stelle seien nur die wichtigsten erwähnt:

- Total verbundenes Netz

Bei einem total verbundenen Netz ist der Ausgang jedes Neuron mir den Eingängen aller anderen Neuronen verbunden. Die Anzahl der Verbindungen bei einem Netz mit K Neuronen ist demnach K x K. Ein solches Netz ist in Bild 3.17 dargestellt

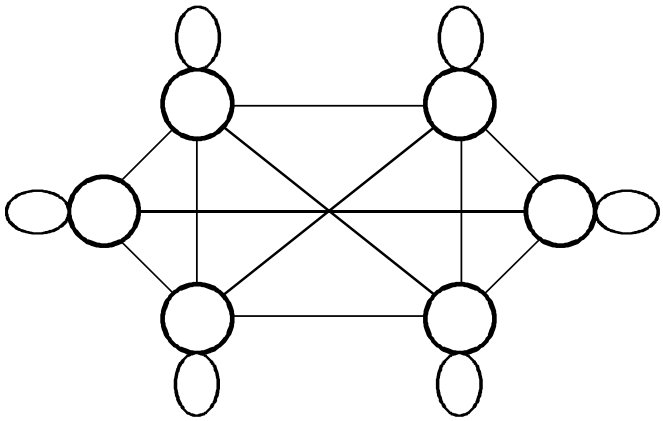


Bild 3.17: Total verbundenes neuronales Netz

- Hierarchisches neuronales Netz

In der Regel sind Neuronale Netzwerke allerdings eher hierarchisch aufgebaut. Ein solches Netz, mit drei Schichten, ist in Bild 3.18 dargestellt.

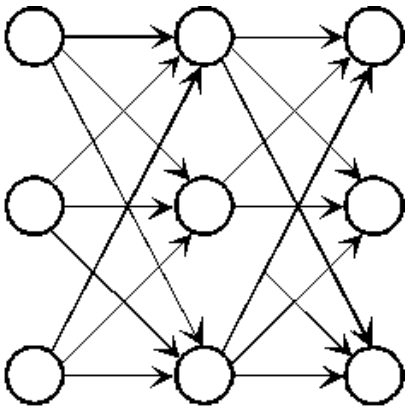


Bild 3.18: Hierarchisches neuronales Netz

Je nach Einsatz kann das neuronale Netz mit oder ohne Rückkopplungen aufgebaut werden.

Man spricht bei Netzen ohne Rückkopplungen von nichtrekursiven oder auch von Feed-Forward-Netzen. Einem solchen Netz entspricht auch das Bild 3.18.

- Rekursives Netz

Man spricht bei Netzen mit Rückkopplungen von rekursiven oder auch von Feed-Back-Netzen. Ein solches Netz ist in Bild 3.19 dargestellt.

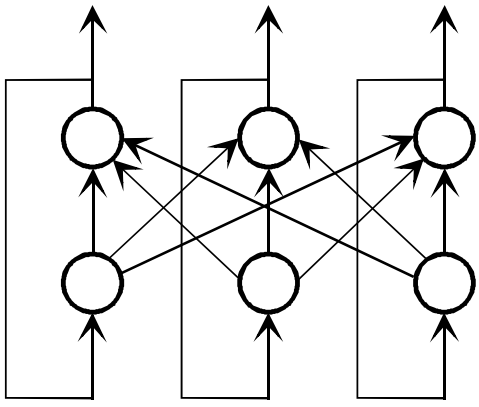


Bild 3.19: Rekursives Netz

Ein rekursives Netz kann im Gegensatz zu nichtrekusiven Netzen, zumindest für eine gewisse Zeit, auch auf frühe Ausgangszustände zurückgreifen.

Wenn ein hierarchisches nichtrekursives Netz außer der Schicht von Eingangsneuronen, die für die Signalverteilung zuständig sind, auch eine Schicht von Rechenneuronen enthält, heißt dieses ein einfaches einschichtiges Netz. Die

Ausgangssignale sind die Funktion der gewichteten Summe der zu dem Eingang des jeweiligen Neurons gelangenden Signale. Im einfachsten Fall stellt der Ausgang einfach die gewichtete Summe der zu dem Eingang jedes Rechenneurons gelangenden Signale dar. Die Gesamtheit der Ausgangssignale bildet den Netzausgangsvektor Y, dessen Dimension m gleich der Zahl der Netzausgänge ist. Definiert man gleichartig den Eingangsvektor X mit der Dimension n und die Matrix W der Gewichtskoeffizienten mit der Dimension $n \times m$, so erhält man die Abhängigkeit des Ausganges des Netzes von dessen Eingang in folgender Vektorform:

$$Y = WX \tag{3.32}$$

Es ist wichtig, dass sich die Rechenkraft derartiger Netze, die eine lineare Aktivierungsfunktion verwenden, bei der Einführung neuer Schichten nicht steigert. Wird z.B. ein zweischichtiges Netz mit den Matrizen der Gewichtskoeffizienten W_1 und W_2 jeweils für die erste und zweite Schicht, betrachtet, so lautet der Ausgangsvektor der Neuronen der ersten Schicht wie folgt

$$Y_1 = W_1X, \tag{3.33}$$

der, der zweiten Schicht wird nach der Formel

$$Y_2 = W_2\, Y_1 = W_2W_1X = WX \tag{3.34}$$

Berechnet. Hierbei ist $W = W_2W_1$ die Matrix der Gewichtskoeffizienten des einschichtigen Netzes.

Auf diese Weise lässt sich jedes mehrschichtige Netz mit linearer Aktivierungsfunktion auf ein äquivalentes einschichtiges Netz zurückführen.

Als deterministisches neuronales Netz wird ein neuronales Netz bezeichnet, das eine deterministische Aktivierungsfunktion besitzt. Als Aktivierungsfunktion können eine Vielzahl von Funktionen verwendet werden. Die am häufigsten benutzten Aktivierungsfunktionen sind:

Lineare Funktion

Sie wird durch die folgende Gleichung beschrieben:

$$y = k\sum_i x_i w_i \quad . \tag{3.35}$$

Die lineare Funktion ist grafisch in Bild 3.20 dargestellt. Für den Fall, dass k=1 ist werden den Neuronenausgängen die gemeinsamen gewichteten Eingänge zugeführt. Der Anwendungsbereich dieser Aktivierungsfunktion ist sehr beschränkt. Bei dem Aufbau mehrschichtiger Netze basierend auf dieser Funktion führt dies zu keiner Steigerung der Rechenkraft, da sich jedes mehrschichtige Netz mit linearen Elementen auf ein einschichtiges zurückführen lässt sie (3.36).

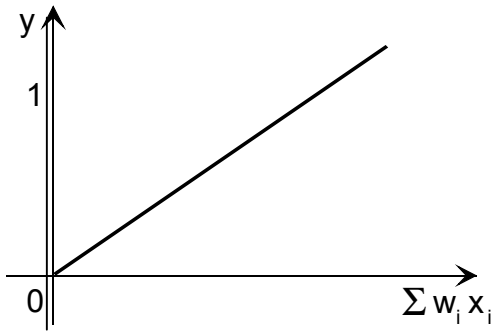


Bild 3.20: Lineare Funktion

Schwellenwertfunktion

Sie wird durch die folgende Gleichung beschrieben:

$$y = \begin{cases} 1, \text{ falls } \sum w_i x_i \geq \theta \\ \\ 0, \text{ sonst} \end{cases} \tag{3.36}$$

bzw.

$$Y = \text{sgn}\{\sum_i x_i w_i - \theta\}$$

Diese Funktion ist grafisch in Bild 3.21 dargestellt. Die Neuronen, welche die Schwellenwertfunktion verwenden, ändern sofort ihren Zustand von „0“ auf „1“,

wenn der Wert des gemeinsamen gewichteten Eingangsvektors den Schwellenwert erreicht.

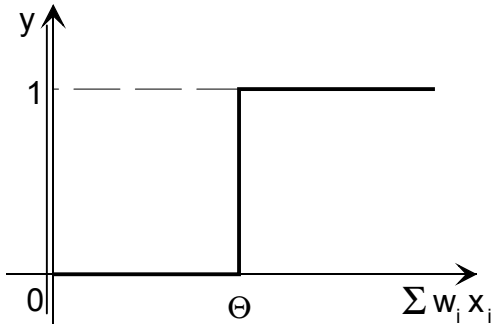


Bild 3.21: Schwellenwertfunktion

Sigmoide (halblineare) Funktion

Sie wird durch die folgende Gleichung beschrieben:

$$y = \frac{1}{1 + e^{-\sum w_i x_i}} \tag{3.37}$$

Diese Funktion ist grafisch in Bild 3.22 dargestellt. Dies ist die am meisten verwendete Aktivierungsfunktion. Strebt der gemeinsame Eingang gegen 0, so strebt der Aktivierungspegel gegen 0. Bei sehr großen Werten am gemeinsamen Eingang ist der Aktivierungspegel praktisch gleich „1“.

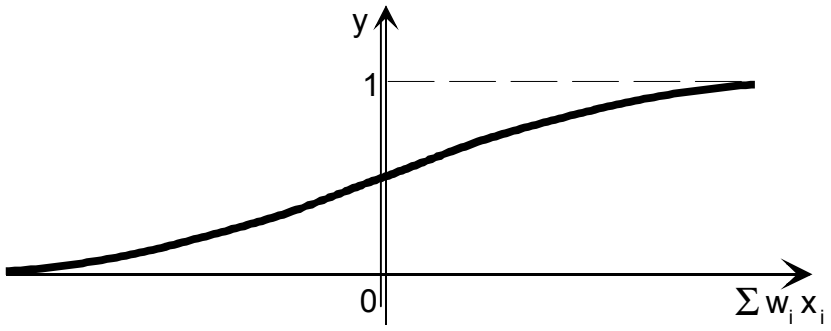


Bild 3.22: Sigmoide (halblineare) Funktion

Hyperbeltangens

Sie wird durch die folgende Gleichung beschrieben:

$$y = th(\Sigma w_i x_i) \tag{3.38}$$

Diese Funktion ist grafisch in Bild 3.23 dargestellt. Im Unterschied zur sigmoiden Funktion schwankt der Aktivierungspegel der Neuronen unter dem Einsatz des Hyperbeltangens von -1 bis +1.

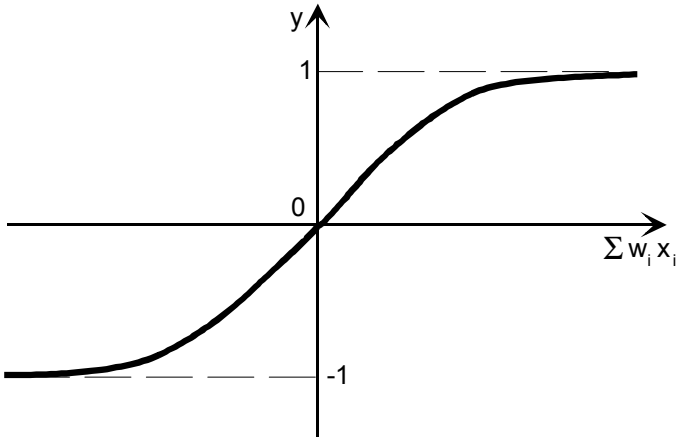


Bild 3.23: Hyperbeltangens

Die drei zuletzt gezeigten Aktivierungsfunktionen sind für eine Vielzahl von Anwendungen relevant. Die Wertebereiche dieser Funktionen [0; 1] oder [-1; 1] bieten eine Vielzahl von Möglichkeiten zur Klassifizierung der Eingangsvektoren.

3.2.3 Lernverfahren in neuronalen Netzen

Bei der Initialisierung eines künstlichen Neuronalen Netzes werden die Netzparameter, wie Gewicht und Schwellenwerte, relativ zufällig gewählt. Hat man schon Wissen über das System kann das an dieser Stelle mit in den Entwurf einfließen. Zu diesem Zeitpunkt kann man von dem System noch keine guten Ergebnisse erwarten. Das System muss erst trainiert werden, es muss lernen.

Das Lernen in neuronalen Netzen kann als Suche nach dem Minimum der mehrdimensionalen Kriterienfunktion (Fehler- oder Energiefunktion) betrachtet werden. Je nach dem Aufgabentyp kommen Suchverfahren zum Einsatz, die in der Lage sind, globale oder lokale Extrema (Flächen im Raum der Gewichte) zu finden. Man unterscheidet bei den Verfahren das Suchen nach lokalen (z.B. Gradientenmethode) Extremstellen, nach dem globalen (z.B. genetische Algorithmen) Extremum und Hybridmethoden welche Kombinationen der

vorherigen Varianten sind. Bild 3.24 [19] zeigt eine Übersicht über die Klassifizierungen der Lernmethoden wie sie bei neuronalen Netzen vorkommen.

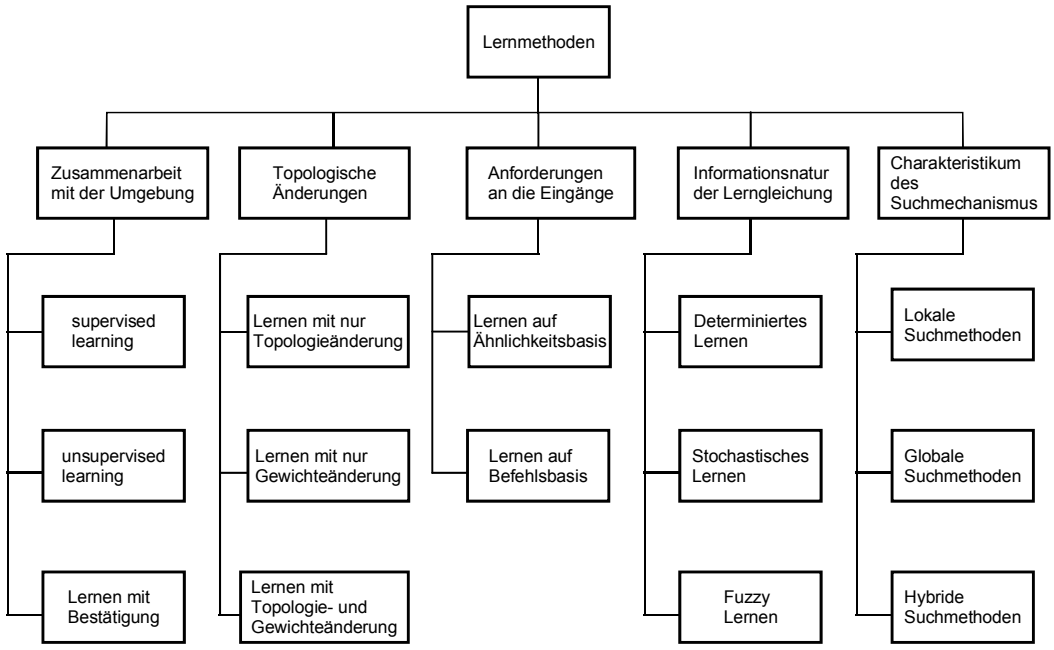


Bild 3.24: Klassifizierung der Lernmethoden in neuronalen Netzen [19]

3.2.3.1 Supervised Learning

Bei der Methode des supervised Learning überprüft ein „Lehrer“ anhand von festen Regeln und Vorschriften den Ausgangsvektor bei einem bekannten Eingang. Diese Paare von Ein- und Ausgangsvektoren nennt man Lernpaare.

Die Netzparameter selber werden mit Hilfe eines iterativen Verfahrens gebildet. Als Startwerte hierfür werden entweder willkürliche Werte verwendet oder mit a-priori Wissen verfeinerte Werte. Bei jedem Iterationsschritt werden aus den Beispieleingängen die aktuellen Ausgänge berechnet. Diese Ausgänge werden dann mit dem gewünschten Ausgang verglichen und der auftretende Fehler ε wird dem „Lehrer“ zu Überprüfung vorgelegt. Er ist dann in der Lage mit Hilfe des vorher festgelegten Lernalgorithmus entsprechende Korrekturen zu veranlassen. Mit Hilfe dieser Änderungen wird dem neuronalen Netz die grundsätzliche Struktur der Wertepaare trainiert. Nach dem Training ist das System dann auch in der Lage mit anderen, nicht trainierten Wertepaaren umgehen zu können und auch akzeptable Ergebnisse zu erbringen.

3.2.3.2 Unsupervised Learning

Bei dieser Lernmethode wird keine explizite Lernüberwachung gemacht. Es ist nur der Eingangs- und Ausgangsvektor vorhanden. Das Ziel, das mit dieser Lernmethode erreicht werden soll, ist es Regelmäßigkeiten bzw. Zusammenhänge in den Eingangswerten zu finden. Hierfür werden die Parameter des Netzes mit bestimmten Methoden modifiziert. Eine der bekanntesten Methoden basiert auf dem „competitive learning Algorithmus“.

Dieses Verfahren hat sehr große Ähnlichkeit mit der Cluster-Analyse. Den Ausgangsneuronen werden hier Cluster zugeordnet. Wenn der Eingangsvektor angelegt wird versucht jedes Neuron in diesem Vektor die ihm eigenen Charakteristiken aufzudecken. Weitergehend findet ein „Wettbewerb“ zwischen den Neuronen in der Form statt, dass das Neuron welches die besten Ergebnisse liefert aktiviert wird. Alle anderen Neuronen gehen in den Zustand „0“über.

Die Hauptanwendung für solche Verfahren sind Feedback-Netze, welche Beispielmuster lernen können. Tiefergehende Informationen sind hiermit nicht zu realisieren. Unbekannte Eingabemuster werden hierbei auf bekannte Ausgangsmuster zurückgeführt. Mann spricht hierbei auch von „Associavite Memory“. Ein weiterer bekannter Lernalgorithmus aus diesem Bereich ist die Hebb´sche Lernregel.

3.2.3.3 Reinforcement Learning

Dieses Lernverfahren hat eine Zwischenposition zwischen den beiden vorangegangenen Verfahren. Das Prinzip folgt dem Ansatz der Belohnung oder Bestrafung seitens des Lehrers. Dabei versucht das System sich so zu modifizieren, dass die Häufigkeit der Belohnung bzw. Bestätigung höher wird. Es ist hierfür allerdings nur ein Informationsbit vorhanden. Es gibt nur an, ob der Status „Gut“ oder „Schlecht“ ist. Informationen über die Güte der Netzfunktionen sind nicht vorhanden. Diese Auswertung erfolgt nach unbekannten Regeln und ohne bekanntes Schema. Die Lernregeln werden während des Lernvorgangs vom System entwickelt.

3.2.4 Lernregeln für neuronale Netze

In den nachfolgenden Punkten werden die wichtigsten Lernregeln für neuronale Netze dargestellt. Die Kombination von Lernregeln, Lernverfahren und der Netzstruktur bildet das neuronale Netz.

3.2.4.1 Delta Regel [52]

Bei der Delta-Regel wird der Fehler ε zwischen dem tatsächlichem Ausgang eines Neurons und dem gewünschten Ausgangswert berechnet. Es erfolgt eine Änderung der Gewichte in den Synapsen, die sich aus den partiellen Ableitungen des Lernfehlers ε nach dem Gewichtungsfaktor bilden und iterativ verbessert werden.

Es wird der Ausgangswert Y mit den zuerst ausgewählten Gewichtungsfaktoren nach der folgenden Formel berechnet:

$$y = \sum_{i=1}^{n} w_i(k) \cdot x_i \tag{3.39}$$

Dieser Wert wird mit dem Sollwert y´ verglichen. Dann wird der quadratische Fehler der beiden Werte nach der folgenden Gleichung berechnet:

$$\varepsilon = (y' - y)^2 = \left(y' - \sum_{i=1}^{n} w_i(k) \cdot x_i \right)^2 \tag{3.40}$$

Danach wird der Gewichtungsfaktor w_i in die Richtung des lokalen Minimums verändert. Dies geschieht nach der folgenden Gleichung:

$$w_i(k) = w_i(k-1) - \beta \cdot \frac{\partial \varepsilon}{\partial w_i} \tag{3.41}$$

mit Verwendung von

$$\frac{\partial \varepsilon}{\partial w_i} = -2 \cdot \left(y' - \sum_{i=1}^{n} w_i(k) \cdot x_i \right) \cdot x_i \tag{3.42}$$

ergibt sich

$$w_i(k) = w_i(k-1) + \beta \cdot (y' - y) \cdot x_i \tag{3.43}$$

Diese Lernregel wird so lange befolgt, bis der Fehler ε gegen Null geht. Diese Lernregel beschreibt die zeitliche Änderung der synaptischen Gewichte.

3.2.4.2 Hebb´sche Lernregel

Einen anderen Lernalgorithmus stellt dieser Algorithmus dar. Er geht davon aus, dass eine Veränderung der Gewichtung in den Synapsen immer proportional zu dem Produkt von Eingangs- und Ausgangssignal ist [19].

Für jede Verbindung von Neuron i nach Neuron j gilt:

$$\Delta w_{ij} = \eta x_i \cdot y_j \tag{3.44}$$

mit η= Lernschrittweite $\eta \in [0,1]$
Es gilt dann nach Hebb:

$$\begin{bmatrix} \Delta w_{11} & \cdot & \cdot & \cdot & \Delta w_{1n} \\ \Delta w_{21} & \cdot & \cdot & \cdot & \Delta w_{2n} \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ \Delta w_{n1} & \cdot & \cdot & \cdot & \Delta w_{nm} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_n \end{bmatrix} * \begin{bmatrix} y_1 & y_2 & \cdot & \cdot & y_n \end{bmatrix} \tag{3.45}$$

Zu den Lernzeitpunkten k=1, 2,...K werden die Ein/Ausgangspaare (x_k, y_k) zu den vorherigen Paaren addiert, wobei als Anfangsbedingung in der Regel w_{ij} =0 gesetzt wird.

So ist die Gewichtungsmatrix w nach Hebb in einem Rechenzug berechenbar. Dies geschieht mit der folgenden Gleichung:

$$w(K) = w = \sum_{u=1}^{K} \underline{x}_u * \underline{y}_u^T \tag{3.46}$$

3.2.4.3 Lernalgorithmus nach Kohonen

Eine der bekanntesten Lernregeln für künstliche Neuronale Netze ist der Lernalgorithmus nach Kohonen [19].

$$w_{ij}(n+1) = w_{ij}(n) + \eta[x_j - w_{ij}(n)] \tag{3.47}$$

Hierbei ist x_j der Eingangsvektor, w_{ij} der Wichtungsvektor und η die Lernrate. Der Abstand D_j im n-dimensionalen Raum wird minimiert mit

$$D_j = \sqrt{\sum_i (x_i - w_{ij})^2} \tag{3.48}$$

3.2.4.3 Backpropagation Algorithmus

Der Backpropagation Algorithmus arbeitet mit der Minimierung des quadratischen Fehlers zwischen dem realen und dem gewünschten Ausgangsvektor. Diese Minimierung erfolgt nach der Gradientenmethode.

Der Backpropagation Algorithmus ist eine Verallgemeinerung der Delta-Regel für mehrschichtige neuronale Netze [52, 53]

Der Backpropagation Algorithmus wird bei Feedforward Netzen eingesetzt. Er ist in der Lage auch mit möglicherweise verdeckten Schichten von Neuronen zu arbeiten.

Es gibt allerdings eine grundlegende Bedingung für die Verwendung des Backpropagation Algorithmus. Sie liegt in der Verwendung der Gradienten-methode und zwar muss die Aktivierungsfunktion vollständig stetig und differenzierbar über den gesamten Wertebereich sein. Aus diesem Grund kommt hierfür in der Regel die Sigmoiden-Funktion zum Einsatz.

Das Ziel dieses Algorithmus ist es auch wieder ein Fehlersignal, hier einen Fehlervektor zu minimieren. Hierfür werden zu Anfang für einen vorgegebenen Eingangsvektor alle Neuronen in Vorwärtsrichtung schichtweise berechnet. Die Eingangswerte werden hierfür zuerst gewichtet und dann aufsummiert. Danach wird die Ausgangsfunktion mit Hilfe der Aktivierungsfunktion erzeugt.

Dieser generierte Ausgangsvektor wird dann mit dem gewünschten Ausgangsvektor verglichen. Man ermittelt hiermit einen Fehlervektor und eine Fehlerrate. Diese Werte werden dann an die Vorgängerschicht zurückgegeben.

Der Fehler wird berechnet mit:

$$Err_j = \frac{1}{2}\left(ys_j - y_j\right)^2 \tag{3.49}$$

hierbei ist ys_j der Sollwert, y_j der Istwert der Ausgangsschicht und j der Index über alle Neuronen der Ausgangsschicht.

Durch das Verändern der einzelnen Verbindungsgewichte w_{ij} ändert sich der Ausgabewert von Neuronen der Ausgabeschicht und damit auch Err_j.

Die Minimierung von Err_j wird über die Verringerung der Lernfehler Ep für jedes Paar angestrebt, dies bedeutet, dass solche w_{ij} bestimmt werden, die die Lernfehler Ep des Paares reduzieren (Gradientenabstiegsverfahren)

$$\Delta w_{ij} = -\varepsilon \cdot \frac{\partial E_p}{\partial w_{ij}} \tag{3.50}$$

(Veränderungen der Verbindungsgewichte sind proportional zur Steigung der Fehlerfunktion, die Lernrate ε ist Proportionalitätsfaktor
Aus Gl. 3.51 folgt mit Hilfe der Kettenregel:

$$\Delta w_{ij} = -\varepsilon \cdot \frac{\partial E_p}{\partial net_i} \cdot \frac{\partial net_i}{\partial w_{ij}} \tag{3.51}$$

Mit der Definition des Fehlersignals für ein Neuron i

$$\delta_i = -\frac{\partial E_p}{\partial net_i} \tag{3.52}$$

und

$$\frac{\partial net_i}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \sum_j y_j \cdot w_{ij} = y_j \tag{3.53}$$

folgt

$$-\varepsilon \cdot \frac{\partial E_p}{\partial net_i} \cdot \frac{\partial net_i}{\partial w_{ij}} = \varepsilon \cdot \delta_i \cdot y_j \tag{3.54}$$

Bei der Verwendung einer linearen Aktivierungsfunktion erhält man die Delta-Regel. Mit einer nicht linearen und differenzierbaren Aktivierungsfunktion kann das Fehlersignal bestimmt werden:

$$\delta_i = (ys_i - y_i) \cdot F'(net_i) \tag{3.55}$$

Für Backpropagation-Netzwerke wird eine sigmoide Abbildungsfunktion gewählt. Hierdurch werden mögliche Aktivierungen auf das Intervall (0,1) beschränkt. Der asymptotische Verlauf der sigmoiden Aktivierungsfunktion stellt sicher, dass sich extremes Verhalten einzelner Neuronen nicht über das ganze Netzwerk fortpflanzen kann.

Mit

$$F'(net_i) = y_i \cdot (1 - y_i) \tag{3.56}$$

und der Identitätsfunktion als Ausgabefunktion ergibt sich

$$\delta_i = (ys_i - y_i) \cdot y_i \cdot (1 - y_i) \tag{3.57}$$

Diese Formel ist nur auf Neuronen der Ausgabeschicht anwendbar, da für innere Neuronen kein konkreter Ausgabewert ys_i vorgegeben ist.
Durch Anwendung der Kettenregel kann der Lernfehler eines inneren Neurons rekursiv aus dem Produkt des Fehlersignals übergeordneter Neuronen und den entsprechenden Verbindungsgewichten bestimmt werden.
Für alle Neuronen der inneren Schicht gilt:

$$\delta_i = \sum_k \delta_k \cdot w_{ik} \cdot F'_i(net_i) \tag{3.58}$$

Die Berechnung des Fehlersignals erfolgt hier entgegengesetzt der vorwärtsgerichteten Propagierung von Eingabesignalen, d.h.:

Zunächst werden die Fehlersignale von Neuronen übergeordneter Schichten berechnet, bevor die Fehlersignale der aktuellen Schicht ermittelt werden können. Wegen dieser der Informationsausbreitung entgegengesetzten Fehler-propagierung, heißt das Verfahren Backpropagation. Mit einer sigmoiden Aktivierungsfunktion wird das Fehlersignal für Neuronen der inneren Schichten durch die Gleichung

$$\delta_i = y_i \cdot (1 - y_i) \cdot \sum_k \delta_k \cdot w_{ik} \tag{3.59}$$

berechnet.
Das Lernverhalten wird durch den Verlauf der Fehlerfunktion entscheidend beeinflußt. Im ungünstigsten Fall konvergiert der Backpropagation-Lernalgorithmus gegen einen Wert, der nur in einem bestimmten Bereich der Fehlerfunktion minimal ist (lokales Minimum). Da stets eine Gewichtsveränderung in Richtung des steilsten Abstiegs der Fehlerfunktion erfolgt, besteht die Gefahr:

1. Der Fehlerwert oszilliert um ein lokales Minimum
2. Dieses Tal der Fehlerfunktion kann nicht mehr in Richtung des globalen Fehlerminimums verlassen werden.

Ein schnelles Absinken des Fehlerwerts kann durch einen möglichst großen Wert der Lernrate (nahe 1) erreicht werden. Dabei kann aber der Fehlerwert über das Minimum der Fehlerfunktion hinauslaufen. Im folgenden Lernzyklus ist dann eine Veränderung des Verbindungsgewichts in genau entgegengesetzter Richtung erforderlich. Ein wiederholtes Auftreten dieses Effekts führt zu unerwünschten Schwingungen um das Minimum der Fehlerfunktion. Das Oszillieren kann durch Berücksichtigung des vorliegenden Werts der Gewichtsveränderung bei der Berechnung des aktuellen Gewichtsveränderungswerts vermieden werden:

$$\Delta w_{ij}(t+1) = \varepsilon \cdot \delta_i \cdot y_j + \alpha \Delta w_{ij}(t) \tag{3.60}$$

Der Wert α gibt an, wie stark der Wert der alten Gewichtsänderung in die Berechnung der neuen Gewichtsänderung einfließen soll. Üblicherweise wird beim Trainieren von Backpropagation-Netzwerken α mit 0.9 und ε nahe dem Wert 0.3 gewählt.

3.3 Neuro-Fuzzy Technologie

Die größten Probleme bei den bisher beschriebenen Systemen liegen in der Aufstellung der Zugehörigkeitsfunktionen und in der Bildung der Wissensbasis bzw. im Aufbau der Struktur des Netzes. Wenn in diesen Bereichen Fehler gemacht werden, bzw. nicht alle Zustände berücksichtigt werden, kann dies zu großen Abweichungen von der möglichen optimalen Lösung führen, oder sogar zu Instabilitäten führen, wenn es sich um Aufgabenstellungen der Regelungstechnik handelt. Für den richtigen Aufbau solcher Systeme benötigt der Entwickler einige Erfahrungen.

Einen Ausweg aus diesen Problemen bieten Neuro-Fuzzy-Systeme. Hierbei übernimmt das System die Aufgabe, die optimalen Zugehörigkeitsfunktionen ebenso wie die optimale Wissensbasis zu bilden. Die bei Neuro-Fuzzy-Systemen eingehenden Werte werden wie bei reinen Anwendungen der Fuzzy-Logik zuerst fuzzifiziert. Die Formen und die Werte der Zugehörigkeitsfunktionen haben in der Lernphase zuerst voreingestellte Werte, welche entweder willkürlich oder nach bestem Wissen erstellt sind. Danach gelangen diese Werte in die weiteren Schichten des Neuronalen Netzes und generieren in den Ausgangs-Neuronen einen scharfen Wert. Wenn dieser nicht bis auf einen vorgegebenen Fehler an den gewünschten Ausgangswert heran reicht, werden die Gewichte des Netzwerkes, mit den unter anderem mit den nachfolgend beschriebenen Verfahren so geändert, bis der gewünschte Ausgangswert generiert wird.

3.3.1 Neuronale Fuzzy Netze

3.3.1.1 Typen neuronaler Fuzzy Netze

Unter einem neuronalen Fuzzy Netz versteht man neuronale Netze, die in der Lage sind, Fuzzy-Signale zu verarbeiten und wobei die Gewichte auch Fuzzy-Gewichte sein können (ist allerdings keine Bedingung, es gibt auch Neuro-Fuzzy Netze mit scharfen Gewichten [19]).

Es werden drei verschiedene Typen von Neuronalen Fuzzy Netzen unterschieden:

Typ 1: Bei Netzen dieses Typs werden scharfe (crispe) Signale X_i (i=1,…, n) verarbeitet. Die Gewichte sind allerdings Fuzzy-Zahlen ($\tilde{w}_{ij}, \tilde{k}_j$). In Bild 3.25 ist ein solches Netz dargestellt.

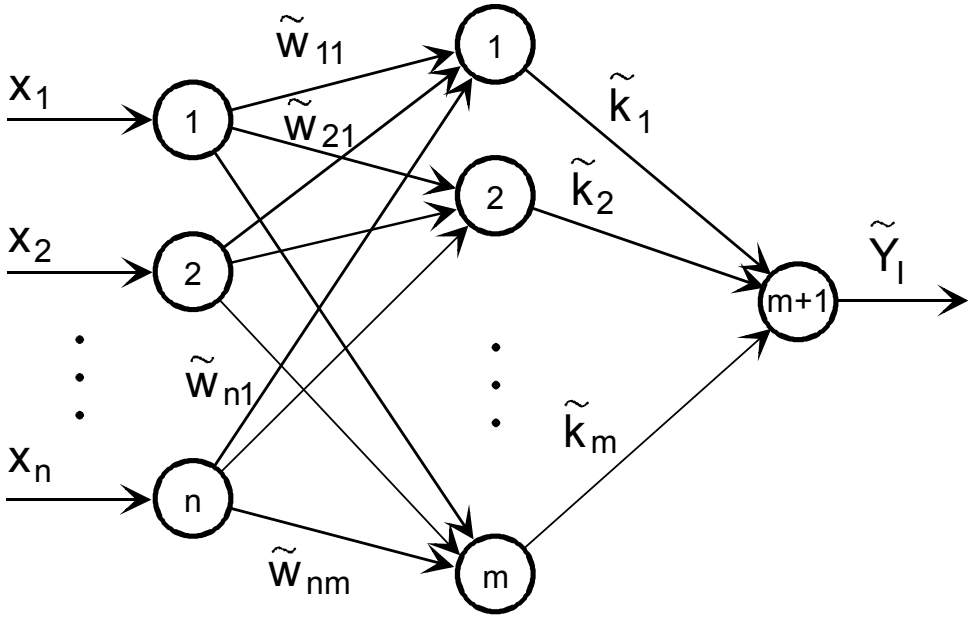


Bild 3.25: Typ 1 neuronaler Fuzzy-Netze

Typ 2: Bei Netzen dieses Typs werden fuzzy Signale $\tilde{X}_i$ (i=1,…, n) verarbeitet. Die Gewichte allerdings sind scharf w_{ij}, k_j. In Bild 3.26 ist ein solches Netz dargestellt.

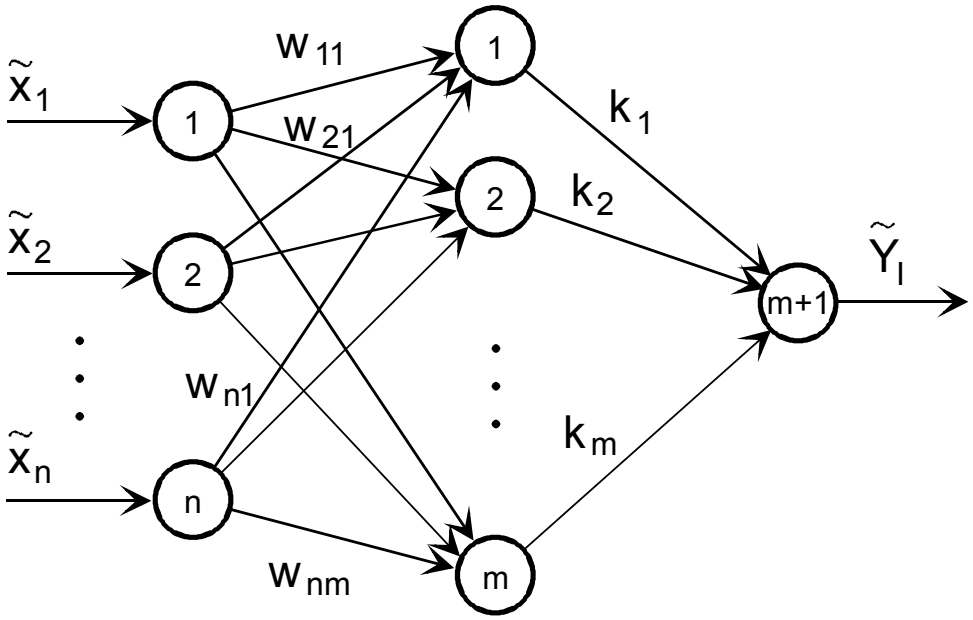


Bild 3.26: Typ 2 neuronaler Fuzzy-Netze

Typ 3: Bei Netzen dieses Typs werden fuzzy Signale $\tilde{X}_i$ (i=1,…,n) verarbeitet. Die Gewichte sind Fuzzy-Zahlen ($\tilde{w}_{ij}, \tilde{k}_j$). In Bild 3.27 ist ein solches Netz dargestellt.

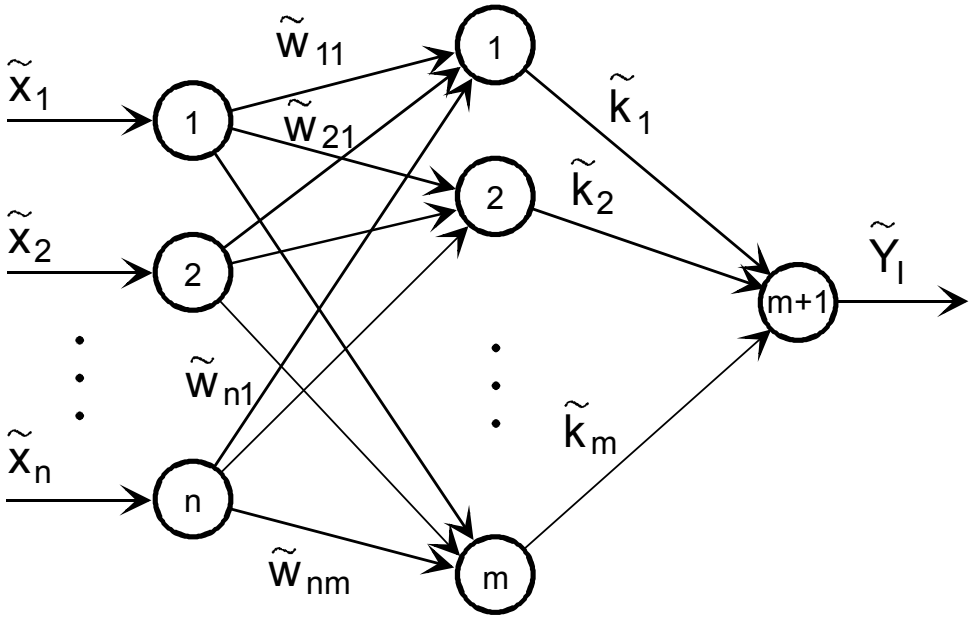


Bild 3.27: Typ 3 neuronaler Fuzzy-Netze

Für die nachfolgenden Betrachtungen werden neuronale Fuzzy-Netze des Typs 3 verwendet.

3.3.1.2 Fuzzyfizierte Lernalgorithmen

Es sind nahezu alle Varianten und Möglichkeiten wie bei „normalen“ Neuronalen Netzen bei Neuro-Fuzzy-Netzen möglich. Teilweise sind nur Anpassungen vorzunehmen.

Nachfolgend ist ein Lernalgorithmus für neuronale Fuzzy-Netze beschrieben [19].

Es sei $\tilde{X}_l = \{\tilde{X}_{l1},...,\tilde{X}_{ln}\}$ ein Vektor von Fuzzy-Mengen, $\tilde{Y}_l^*$ sei der gewünschte Ausgang und $\tilde{Y}_l$ der aktuelle Ausgang des Netzes. Alle Signale $\tilde{X}_l$, $\tilde{Y}_l$, $\tilde{Y}_l^*$ werden als Fuzzy-Mengen aus [0, 1] angenommen. Die Gewichte W_{ij} und K_j sind beliebige Fuzzy-Mengen reeller Zahlen. Es gilt:

$$\widetilde{Y}_l = f\left[\sum_{j=1}^{m} \widetilde{K}_j \widetilde{Z}_{lj}\right], \tag{3.61}$$

hierbei ist $\widetilde{Z}_{lj}$ der Ausgang des j-ten Neurons der verdeckten Schicht welche $\widetilde{X}_l$ als Eingang hat. Es gilt weiterhin:

$$\widetilde{Z}_{lj} = f\left[\sum_{i=1}^{n} \widetilde{X}_i \widetilde{W}_{ij}\right] \tag{3.62}$$

Als Aktivierungsfunktion ist die Sigmoid-Funktion ausgewählt.

Die Fuzzy-Entfernung zwischen dem Sollwert und dem reellen Ausgang des Netzes wird bestimmt mit:

$$\widetilde{E} = \frac{1}{2}\sum_{i=1}^{n} (\widetilde{Y}_i - \widetilde{Y}_i^*)^2 \tag{3.63}$$

Der Sinn des Algorithmus besteht darin, Werte für $\widetilde{W}_{ij}$ und $\widetilde{K}_j$ zu finden, die $\widetilde{E}$ gegen 0 konvergieren lassen.
Für den Fall, dass $\widetilde{E}$ nicht zu Null wird, muss eine Stop-Regel eingeführt werden, die einen Abbruch unter der Voraussetzung ermöglicht, dass der Sollwert $\widetilde{Y}_i^*$ in dem Intervall $[y_{i1}, y_{i2}]$, $1 \le i \le n$ liegt. Wenn $\widetilde{Y}_i = \widetilde{Y}_i^*$ für alle i ist, liegt der Grenzwert von $\widetilde{E}$ in dem Intervall $[-\lambda, \lambda]$. Dann gilt:

$$\lambda = \frac{1}{2}\sum_{i=1}^{I} (y_{i2} - y_{i1})^2 . \tag{3.64}$$

Es sei $\varepsilon > 0$ eine akzeptable Abweichung von $\widetilde{E}$, wenn $\widetilde{Y}_i = \widetilde{Y}_i^*$ für alle i gilt. Dann ist die Stop-Regel, die das Ende der Berechnungen von Werten für die Gewichte festlegt, gegeben, wenn $\widetilde{E}$ in dem Bereich Ω liegt:

$$\Omega = [-\lambda-\varepsilon, \lambda+\varepsilon] \times [0, 1] . \tag{3.65}$$

Der Lernalgorithmus für den betrachteten Fall lautet:

$$\tilde{K}_j(q+1) = \tilde{K}_j(q) + \eta\, \Delta\tilde{K}_j(q),\ 1 \le j \le m;\ q=1,2,... \tag{3.66}$$

Es gilt hierbei:

$$\Delta\tilde{K}_j = \frac{\partial\tilde{E}}{\partial\tilde{K}_j} \tag{3.67}$$

und

$$\frac{\partial\tilde{E}}{\partial\tilde{K}_j} = \sum_{i=1}^{L}(\tilde{Y}_i - \tilde{Y}_i^*)(\tilde{Y}_i)(1-\tilde{Y}_i)\,\tilde{Z}_{ij}\ . \tag{3.68}$$

Hierbei ist η die Lernrate.

Die Gewichte W_{ij} berechnet man mit:

$$\tilde{W}_{ij}(q+1) = \tilde{W}_{ij}(q) + \eta\, \Delta\tilde{W}_{ij}(q) \tag{3.69}$$

$$\frac{\partial\tilde{E}}{\partial\tilde{W}_{ij}} = \sum_{i=1}^{L}(\tilde{Y}_i - \tilde{Y}_i^*)(\tilde{Y}_i)(1-\tilde{Y}_i)\,\tilde{K}_j\frac{\partial\tilde{Z}_{lj}}{\partial\tilde{W}_{ij}} \tag{3.70}$$

$$\frac{\partial\tilde{Z}_{lj}}{\partial\tilde{W}_{ij}} = (\tilde{Z}_{lj})(1-\tilde{Z}_{lj})\tilde{X}_{li} \tag{3.71}$$

3.3.1.4 Architektur von Neuro-Fuzzy-Systemen

Am besten kann man den grundsätzlichen Aufbau an Hand von Zeichnungen zwei erklären. In Bild 3.28 sind die Struktur und die Hauptfunktionen des Neuro-Fuzzy-Systems dargestellt. Des Weiteren ist in Bild 3.29 ist der innere Aufbau des Neuro-Fuzzy-Systems dargestellt. Wie hier zu sehen ist, werden im System die Fuzzy-Regeln und die Zugehörigkeitsfunktionen erzeugt [19]. Dies geschieht über die Änderung der einzelnen Gewichtsfunktionen und Schwellenwerte.

Die Startwerte des Systems werden entweder aus der Erfahrung heraus oder auf Grund mathematischer bzw. physikalischer Formeln oder durch die Befragung eines externen Experten realisiert.

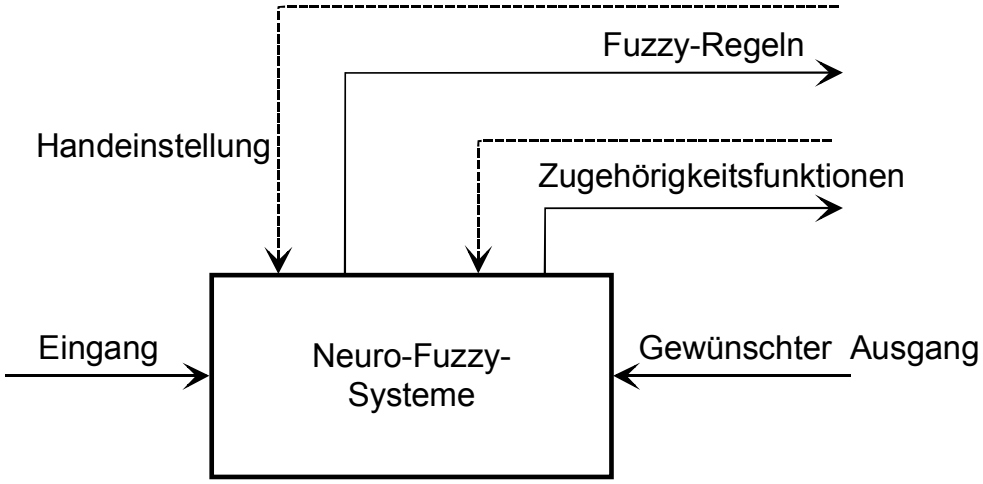


Bild 3.28: Struktur und Hauptfunktionen des Neuro-Fuzzy-Systems

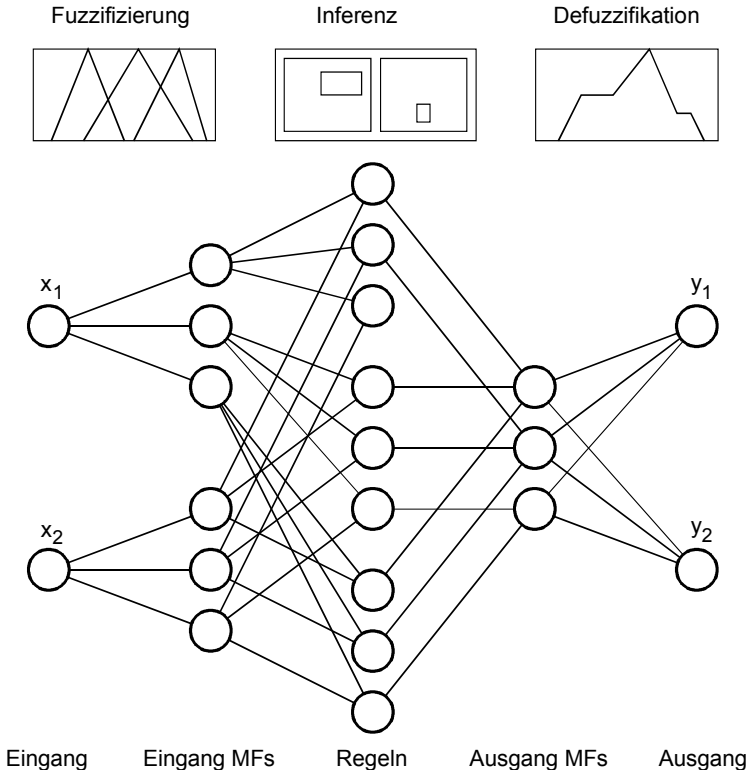


Bild 3.29: Innerer Aufbau des Neuro-Fuzzy-Systems

3.3.1.5 Generierung von Regeln und Zugehörigkeitsfunktionen

Der Vorteil der Realisierung der Ein-/Ausgangs-Relation ist es, dass sich hieraus heuristische Kenntnisse und qualitative Information über das System ermitteln lassen.

Man versteht unter der Generierung der Regeln die Möglichkeit die Parameter und die Struktur eines aufgebauten Netzes in einen Satz von Fuzzy-Regeln mit den gewünschten Bedingungen sowie einem Defuzzyfizierungs-Algorithmus abzubilden. Hierbei wird auch der Satz für die Zugehörigkeitsfunktionen festgelegt [19].

Es werden in vielen Fällen besondere Typen von Neuronen eingesetzt. Sie sind in der Lage die Operationen auszuführen, die für Fuzzy-Logik typisch sind (MIN, MAX, Vereinigung usw.). Dies wiederum erleichtert die Transformation von Regeln in ein Neuro-Fuzzy-Netz sowie die Rückabbildung.

Die Extraktion von Regeln aus einem gelernten Netz ist in der Regel einfach. Die maximale (ausreichende) Anzahl von Neuronen in der 3. Schicht, ist gleich dem Produkt aus der Anzahl von Neuronen in allen Gruppen der 2. Schicht (siehe Bild 3.28). In diesem Fall, kann man höchstens 9 (3×3) Neuronen in der 3. Schicht haben. Dies entspricht der Zahl von Regeln, die man durch Vereinigung aller möglichen Kombinationen für jede Eingangsvariable erhalten kann (vollständig beschriebenes Fuzzy-System). In Übereinstimmung mit den Regeln der Fuzzy-Verarbeitung führen die Neuronen der 3. Schicht die Operation MIN (oder Multiplikation) an allen ihren Eingängen aus.

Für die automatische Generierung der Fuzzy-Regeln wird ein spezieller Algorithmus verwendet. Zuerst wird eine maximal mögliche Anzahl von Regeln generiert. Danach werden auf Grund der Gewichtskoeffizienten bei den Verbindungen unwichtige Regeln sowie die Größe des Ausgangsfehlers extrahiert.

Die Parameter der Neuronen in den entsprechenden Schichten des neuronalen Netzes, die man nach dem Lernen erhält, bestimmen die Zugehörigkeitsfunktionen direkt.

In einem anderen Fall, wenn die Regelschicht auf der Basis eines traditionellen rückkopplungsfreien (feed forward) Back-Propagation neuronalen Netzes verwirklicht wird (diese Funktion wird durch eine oder zwei aufeinander folgende Neuronenschichten mit sigmoider Aktivierungsfunktion realisiert), erfolgt die Extraktion der Regeln durch die Zuführung des Wertes „1.0“ zu den Eingängen für die Regeln.

Eine Vollautomatisierung ist nicht realisierbar. Das Netz ist nicht in der Lage z.B. die erforderlichen linguistischen Namen für Fuzzy-Variable vom Typ „wenig“ oder. „etwas wärmer“ richtig auszuwählen. Da diese Begriffe von Menschen verwendet werden, ist es unbedingt notwendig, dass auch ein Mensch deren Generierung ausführt.

3.3.1.6 Fuzzyfizierung und Defuzzyfizierung in Neuro-Fuzzy-Systemen

Die Besonderheit der Fuzzyfizierung und Defuzzyfizierung in Neuro-Fuzzy-Systemen besteht darin, dass hier das Einstellen oder Lernen eine wichtigere Rolle spielt [19]. Ein Fuzzyfikator hat für einen konkreten Eingang mehrere Ausgänge, wobei jeder Ausgang für einen bestimmten Fuzzy-Begriff steht und er bestimmt zu welchem Grad der Eingang zu ihm gehört.

Die Berechnung einer, der Fuzzy-Größe am Eingang entsprechenden, Defuzzy-Größe ist das Aufgabenfeld eines Defuzzyfikators. Es gibt mehrere Formeln zur Defuzzifizierung, je nach dem, wie der Mittelwert für den Bereich unter der Zugehörigkeitsfunktion am Ausgang zu berechnen ist. Der Vorteil der Defuzzy-fizierung in Neuro-Fuzzy-Systemen besteht darin, dass die Defuzzyfizierungs-formel hier wie die Zugehörigkeitsfunktion darstellbar und modifizierbar ist.

Die in der Regel verwendete Methode zur Defuzzifizierung, die so genannte Massenschwerpunktsmethode lautet:

$$y_{defuz} = \frac{\int y\mu(y)dy}{\int \mu(y)dy} \quad . \tag{3.72}$$

Die Zugehörigkeitsfunktion am Ausgang soll aus einer Kombination der Zugehörigkeitsfunktionen am Eingang bestimmt werden, also kann die

Kombination einiger Formen (z.B. Glocken, Dreiecke) die Eigenschaften der Originalform (Glocke) nicht nachbilden, es lässt sich die resultierende defuzzyfizierende Funktion zu einem einfachen Ausdruck verallgemeinern:

$$y_{defuz} = \frac{\sum_i y_i \mu(y)}{\sum_j \mu(y_j)} \quad . \tag{3.73}$$

Hierbei steht μ für eine Funktion, die stetig und stückweise differenzierbar ist.

In vielen Fällen werden die Zugehörigkeitsfunktionen am Ausgang als Singletons ausgedrückt. In diesem Fall wird der Defuzzyfikator auf Basis des folgenden Neurons

$$y = \frac{\sum_i x_i w_i}{\sum_j w_j} \tag{3.74}$$

aufgebaut.

3.4 Genetische Algorithmen

Genetische Algorithmen verhalten sich wie Gene in der Natur. Sie teilen sich, bilden sich neu oder mutieren auch. Das Wichtigste ist aber, dass die Besten und Fittesten „überleben“. In dieser Anwendung steht ein Gen für eine bestimmte Anzahl von Gewichten in dem Neuronalen Netz. Hiermit lassen sich nun die notwendigen Gewichte der Neuronalen Netze so einstellen, dass eine optimale Lösung erreicht wird und, dass dieses Verfahren sehr schnell diese optimale Lösung liefert. Der grundsätzliche Aufbau eines einfachen Genetischen Algorithmus hat die gleiche Struktur [25] wie auch bei anderen evolutionären Programmen. Diese Struktur ist in Bild 3.30 dargestellt.

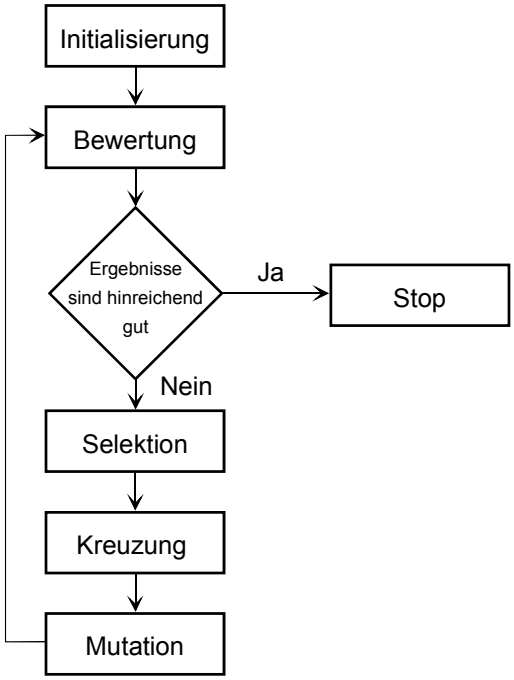


Bild 3.30: Struktur eines einfachen Genetischen Algorithmus

Nachfolgend nun ein Beispiel über die Verhaltensweise von genetischen Algorithmen:

Der Adaptions-Algorithmus soll die Variablen einer Funktion einstellen. Hierfür wird die folgende Prozedur verwendet.

a) Codierung der charakteristischen Punkte in z.B. einen Binär Wert

(0 0 0 0 0)
(0 0 0 0 1)
(0 0 0 1 0)
⋮
(1 1 1 1 1)

b) Es wird eine anfängliche Population erzeugt. Sie kann entweder gezielt eingestellt werden, wenn genug Wissen im Voraus vorhanden ist, oder willkürlich gewählt werden, es gilt n für die Anzahl der Individuen

x_1 = (0 1 1 0 0)
x_2 = (1 1 0 0 1)
⋮
x_n = (0 0 0 1 1)

c) Überprüfung der Funktion mit den Werten der Population

$f(x_1), f(x_2),...,f(x_n)$

d) Verwendung der Fitnessfunktion. Es wird die folgende Fitnessfunktion verwendet, welche der Genetische Algorithmus bei jeder Berechnung benötigt, dies können Energiefunktionale oder auch die Berechnungsvorschrift für die kleinsten Fehlerquadrate sein:

$$F = 0{,}5 \sum_{k=1}^{m} [y_i(k) - y_i^m(k)]^2 \qquad (3.75)$$

hierbei ist y_i der gewünschte Wert und y_i^m der ermittelte Wert mit dem jeweiligen Individuum.

e) Nachdem die Fitness aller Individuen und die Fitness der Population bewertet sind, wird die Wahrscheinlichkeit für das Überleben, bzw. Weiterexistieren jedes Individuum wie folgt berechnet:

$$P_s^1 = y(x_1)/y \qquad (3.76)$$

⋮

$$P_s^n = y(x_n)/y \qquad (3.77)$$

Danach wird die kumulative Wahrscheinlichkeit für jedes Individuum berechnet.

$$P_{cum}^{1} = P_s^{1} \tag{3.78}$$

$$P_{cum}^{2} = P_s^{1} + P_s^{2} \tag{3.79}$$

Auf Grund dieser Beurteilungen wird festgestellt welche Individuen „am Leben“ und in die nächste Generation kommen und welche auf der Strecke bleiben [19].

Die Überlebenden gelangen zum zweiten Hauptoperator genetischer Algorithmen, der Kreuzung.

Hiefür werden aus der Population zufällig einige oder alle Chromosomen gewählt. Deren Anzahl hängt von der ursprünglich angegebenen Wahrscheinlichkeit P_c der Kreuzung ab.

Ist P_c = 0,5 und die Größe der Population = 40, so bedeutet dies, dass in jeder Generation genau zwanzig Chromosomen paarweise gekreuzt werden.

Dies wird wie folgt durchgeführt: Sind X_m und X_k für die Kreuzung zufälligerweise gewählt, und ist die Länge des Chromosoms gleich l, so gibt der Zufallszahlgenerator eine ganze Zufallszahl j, welche die Ungleichung $1 \leq j < l$ erfüllt.

Es sei

X_m = (1 1 0 1 1 0 1)
X_k = (1 1 1 0 0 1 0)

mit j = 3

Dies bedeutet, dass die Nachfahren X_m und X_k nach der Kreuzung wie folgt aussehen werden:

X'_m = (1 1 0 0 0 1 0)
X'_k = (1 1 1 1 1 0 1)

Die Zahl j heißt Kreuzungspunkt. Bei diesem Punkt werden die Chromosomen abgeschnitten und die Schnittstücke getauscht.

Dies erzeugt wiederum eine gewisse neue Anzahl von Chromosomen. Sie werden entweder Nachkommen oder Kinder genannt.

Alle „Überlebenden“ und die neu gebildeten Nachkommen ergeben die neue Population.

Ist dies ausgeführt wird die dritte Hauptoperation bei genetischen Algorithmen verwendet. Es ist die Mutation. Mutation bedeutet in diesem Fall in Analogie zu der in der Natur auftretenden Mutation die Änderung eines Bits im Chromosom. Wenn das Chromosom

$$X_k = (1\ 1\ 1\ 0\ 0\ 1\ 0)$$

für die Mutation ausgewählt ist, so wird die Zufallszahl j, deren Wert kleiner als die Bitzahl des Chromosoms sein muss, erneut verwendet und gibt die Stelle des Gens an, das seinen Wert von „0“ auf „1“ oder umgekehrt ändert. Es sei j = 1. Dies bedeutet, dass das erste Gen des Chromosoms X_k seinen Wert von „1“ auf „0“ ändert, d.h.,

$$X_k' = (0\ 1\ 1\ 0\ 0\ 1\ 0)$$

e) Neujustierung der Funktionen. Der Genetische Algorithmus versucht weiter die Parameter so zu verändern, dass (3.76) minimiert wird. Der Genetische Algorithmus generiert diese Parameter weiter mit Hilfe der oben angegebenen Prozedur. Wenn die Anzahl der Generationen über die vorher eingestellte Grenze kommt (z.B. 150) wird der Genetische Algorithmus abgebrochen sonst wird eine neue Population (NP) erzeugt;

Durch diese Prozedur ist es ohne weiteres möglich auch Funktionen und Verläufe zu optimieren die unstetig sind, da „Sprünge“ in den Funktionen nicht ins Gewicht fallen.

Ebenso ist man mit Genetischen Algorithmen in der Lage auch die globalen Minima zu ermitteln. Der Grund hierfür liegt unter anderem in der Mutation. Die

Optimierung „hängt“ nicht nur lokal an einem Minimum. Sie kann auch in andere Bereiche der Funktion „schauen“ ob hier vielleicht ein noch besserer Wert liegt.

Aus diesem Grund ist hiermit auch eine gute Optimierung von Neuro-Fuzzy-Netzen möglich. Hierfür werden die Gewichtungsfaktoren und Parameter binär codiert. Als Beispiel sei hier die einfachste Variante erwähnt:

$$(00000001001)_2 = 9_{10}$$

Dies bedeutet nichts anderes, als dass man für den dezimalen Wert 9 den binären Wert 00000001001 schreiben kann.

4 Verwendete Methoden und Vereinbarungen

4.1 Zugehörigkeitsfunktionen und Regeln

Es muss bei der Verwendung von Zugehörigkeitsfunktionen darauf geachtet werden, dass die einzelnen Kurvenzüge oder Zugehörigkeitsfunktionen mathematisch gut beschreibbar sind und auch für spätere Veränderungen gut zu verwenden sind. Eine sehr gute Form für die Zugehörigkeitsfunktionen sind die Trapez- oder Dreiecksfunktionen. Hierbei ist anzumerken, dass es sich bei den Dreiecksfunktionen eigentlich um eine Sonderform der Trapezfunktionen handelt. Für den Fall, dass zwischen den Eckpunkten die Länge ≡ 0 beträgt.

Das hier entworfene Programm ermittelt, ausgehend von einer standardisierten Trapezfunktion, mit Hilfe von Fallunterscheidungen die Zuweisung des Wertes zu dem zugehörigen Kurvenabschnitt der Zugehörigkeitsfunktion und berechnet dann den Zugehörigkeitswert.

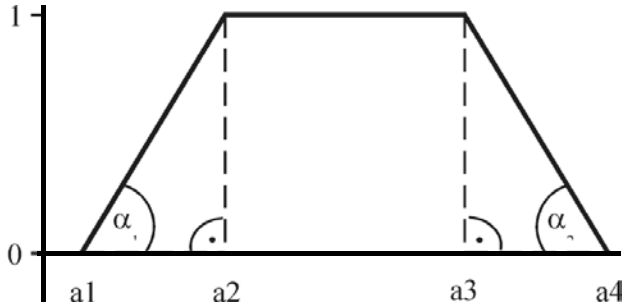


Bild 4.1: Standardisierte Trapezfunktion

Die standardisierte Trapezfunktion, die vom Programm verwendet wird, bildet die Grundfunktion, aus der sich alle anderen verwendeten Kurvenformen der Fuzzy-Sets (n) ableiten. Sie wird über die vier Kurven- oder Eckpunkte, a1 bis a4, definiert. Allgemein muss das Folgende für diese Punkte gelten:

$$a_1 \leq a_2 \leq a_3 \leq a_4$$

Berechnet werden die Zugehörigkeitswerte stückweise abhängig von den Bereichen des Eingangswertes (a_x):

1. Fall : $a1 \le ax \le a2$

$$\Rightarrow \quad y_n(ax) = \tan(\alpha_1) \cdot (ax - a1) = \frac{(ax - a1)}{(a2 - a1)} \tag{4.1}$$

2. Fall : $a2 \le ax \le a3$

$$\Rightarrow \quad y_n(ax) = 1 \tag{4.2}$$

3. Fall : $a3 \le ax \le a4$

$$\Rightarrow \quad y_n(ax) = \tan(\alpha_2) \cdot (a4 - ax) = \frac{(a4 - ax)}{(a4 - a3)} \tag{4.3}$$

4. Fall $ax \le a1$ oder $ax \ge a4$

$$\Rightarrow \quad y_n(ax) = 0 \tag{4.4}$$

Die Darstellung der Werte erfolgt innerhalb der Programme nach dem folgenden Schema. Dies hat den Vorteil, dass trotz bleibender Übersichtlichkeit eine mathematische Weiterverarbeitung möglich ist. Anstelle der kursiv geschriebenen Worte werden wie im nachfolgenden Beispiel für die Fahrbahntemperatur die jeweiligen Werte eingesetzt

=Nummer der ZF==========

Linguistische Variable

Einheit /Wertebereich

Anzahl der einzelnen Sets

Auflistung der einzelnen Sets nach dem folgenden Muster

Name Abkürzung Anzahl der Stützstellen Abszissenwert Zugehörigkeitswert

Beispiel:

=0==========

Fahrbahntemperatur

GradC -30 60

5

sehr_kalt SK 4 -30 1 -7.5 1 -5 0 60 0

kalt K 5 -30 0 -8 0 -2.5 1 1 0 60 0

null N 5 -30 0 -3.5 0 -1 1 1 0 60 0

warm W 5 -30 0 0 0 1.5 1 8 0 60 0

sehr_warm SW 4 -30 0 5 0 7.5 1 60 1

Die Wissensbasen werden mit IF-Then Regeln und Gewichtungsfaktoren aufgebaut. Die Gewichtungsfaktoren g_i mit $i= \overline{1,n}$ können im Wertebereich von 0.1 bis 1 liegen und dienen zur besseren Anpassung bzw. Adaptierung der Wissensbasis.

Ein typischer Aufbau sieht wie folgt aus

WENN Fahrerverhalten ist durchschnittlich UND Bewegung des Gaspedals ist schnell UND Gas wenig DANN Verkehr zähfliesend 1.0

…

WENN Fahrerverhalten ist hektisch UND Bewegung des Gaspedals ist schnell UND Gas wenig DANN Verkehr stockend 0.8

…

WENN Fahrerverhalten ist durchschnittlich UND Bewegung des Gaspedals ist normal UND Gas viel DANN Verkehr flüssig 1.0

…

WENN Fahrerverhalten ist langsam UND Bewegung des Gaspedals ist schnell UND Gas wenig DANN Verkehr zähfliesend 0.2

Das Ergebnis der Inferenz dieser Regeln wird mit dem jeweiligen Gewichtungsfaktor multipliziert und dies wertet die einzelne Regel entweder auf oder ab.

4.2 Adaption der Wissensbasis

Das Ziel dieser Adaption ist es eine hinreichend genaue fuzzy technische Beschreibung des Systems zu geben. Es soll die Benutzer in die Lage versetzten das System so zu beschreiben, wie es Ihnen möglich ist, ohne sich Gedanken darüber machen zu müssen, das es Bereiche gibt, die Aufgrund einzelner Regeln, so beschrieben sind, dass diese Stellen ein Übergewicht im Ergebnis haben. Die Gewichtung der einzelnen Regeln wird zu Beginn für alle Werte auf 1.0 gesetzt, es sei denn es sind Gründe dafür bekannt, dies nicht zu tun, da man im voraus schon weiß, dass diese Regel nicht zu dominant werden darf. In diesem Fall wird auch der Algorithmus diese Regel nicht über den eingestellten Wert vergrößern.

Für die Optimierung bzw. Adaption wird ein zusätzliches System im Parallelbetrieb aufgebaut. Dieses System bekommt genau die gleichen Werte zugeführt wie das Originalsystem. In diesem System werden dann die aktiven und die das Ergebnis erzeugenden Regeln markiert und herausgestellt.

Es wird überprüft wo sich der Träger (siehe Gl 3.8) des Ausgangswertes befindet. Ebenso wird die Distanz zwischen dem ermittelten Ausgangswert und dem tatsächlichen Ausgangswert als Abweichung ermittelt. Ist dieser Fehler gleich Null oder kleiner als ein vorher angegebenes ε ist die Optimierung beendet.

Ist dieser Wert größer, setzt die Optimierung ein. Hierfür werden die n aktivsten Regeln jedes Sets betrachtet (in dieser Arbeit wurde mit n=3 gearbeitet). Sind die Inferenzwerte der n Regeln um mehr als den Wert β verschieden (in dieser Arbeit wurde mit β=0.3 gearbeitet) werden die Regeln, die unterhalb dieser Schwelle liegen, für die Optimierung nicht verwendet.

Diese Filterung ist notwendig, um die tatsächlichen das Ergebnis verursachenden Regeln zu finden. Die Regeln mit dem höchsten Erfüllungsgrad werden in Richtung zu dem richtigen Ausgangswert hin korrigiert. Es sind hierbei zwei Fälle zu betrachten.

1. Fall : Der Träger der Regel mit dem höchsten Erfüllungsgrad aller Regeln liegt über dem Sollwert, das Ergebnis ist nicht optimal

Hierbei wird proportional zur Entfernung des Sollwertes und des Trägers des Ausgangswertes der Wert des Gewichts verkleinert. Das bedeutet, dass wenn z.B. eine linguistische Ausgangsvariable aus fünf Termen besteht und das Ergebnis im Bereich des Trägers des ersten Sets liegt, die Regeln, die ein Ergebnis im fünften Set erzeugen stärker verändert werden (hier niedriger gewichtet) als eine Regel im zweiten Set. Die maximale Schrittbreite γ um die die Gewichtung verändert wird, wird vorgegeben (in dieser Arbeit $\gamma = 0{,}15$). Die Verteilung wurde für diesen Fall auf eine einfache Art und Weise realisiert. Für die hier gemachten Untersuchungen war dies auch ausreichend wie in dem nachfolgenden Kapitel gezeigt. Die Werte für die Schrittbreite γ_i werden berechnet mit:

$$\gamma_i = \frac{\gamma}{m} \cdot k \tag{4.5}$$

hierbei steht γ_i für den Schritt des i-ten Sets, m für die Anzahl aller Fuzzy-Sets und k für den Abstand des Sollwertes zu dem Träger des Ausgangswertes.

Hierdurch wird das neue Gewicht der Regeln zu

$$g_i = -\gamma_i + g_0 = -\frac{\gamma}{m} \cdot k + g_0 \tag{4.6}$$

Hierbei steht g_0 für den bisherigen Gewichtungsfaktor.

Für die Regeln die im Bereich des Sollwertes liegen, wird die Gewichtung nicht geändert. Der Wert für die Gewichtung darf aber nicht unter einen bestimmten Grenzwert fallen. Sollte die Optimierung Werte liefern die dies verursachen, werden ähnlich wie im Fall zwei die Gewichtungen der „richtigen“ Regeln erhöht.

2. Fall : Der Träger der Regel mit dem höchsten Erfüllungsgrad aller Regeln liegt nicht über dem richtigen Ergebnis

Hierbei wird proportional zur Entfernung des Trägers des Ausgangswertes der Wert des Gewichts vergrößert. Das bedeutet, dass wenn eine linguistische Ausgangsvariable aus fünf Termen besteht und das Ergebnis im Bereich des Trägers des ersten Sets liegt die Regeln, die ein Ergebnis im fünften Set erzeugen stärker verändert werden (hier höher gewichtet) als eine Regel im zweiten Set. Die

maximale Schrittbreite γ wird vorgegeben (in dieser Anwendung γ = 0,15). In diesem Fall werden allerdings nur die Regeln verändert, deren Ausgangsträger über dem gewünschten Wert liegen. Die Berechnung erfolgt analog zu Fall 1.

$$\gamma_i = \frac{\gamma}{m} \cdot k \tag{4.7}$$

hierbei steht γ_i für den Schritt des i-ten Sets, m für die Anzahl aller Fuzzy-Sets und k für den Abstand des Sollwertes zu dem Träger des Ausgangswertes.

Hierdurch wird das neue Gewicht der Regeln zu

$$g_i = +\gamma_i + g_0 = +\frac{\gamma}{m} \cdot k + g_0 \tag{4.8}$$

Für die Regeln jedoch, die in den „falschen“ Bereichen liegen, wird die Gewichtung nicht geändert.

Für den Fall, dass Gewichte den Wert 1,0 erreichen, werden ähnlich wie im Fall eins die Gewichtungen der „falschen“ Regeln verkleinert (Gl 4.6)

Nachdem diese Änderungen gemacht wurden, wird derselbe Vektor nochmals verwendet. Dies geschieht so oft bis die Durchläufe abgebrochen werden.

Dieses Verfahren kann durch insgesamt drei Gründe abgebrochen werden:

1. Der Fehler ist gleich Null oder kleiner gleich ε

2. Die vorher angegebene Zahl von Durchläufen ist erreicht

3. Alle Regeln sind an Ihren Grenzen angelangt

Der dritte Grund erzeugt eine Meldung und man müsste entweder von Hand eingreifen oder einen anderen zusätzlichen Durchlauf generieren, der nach zusätzlichen Regeln für die Wissensbasis sucht (siehe Kap 5.4).

Die ermittelten Werte müssen danach noch bei zwei anderen Eingangswerten gute Ergebnisse bringen oder zumindest keine Verschlechterung erzeugen und werden danach in die Wissensbasis des Systems überspielt.

4.3 Lernen in neuronalen Fuzzy-Netzen unter dem Einsatz von α-Schnitten und genetischen Algorithmen

Hier wird nun ein 3-schichtiges rückkopplungsfreies neuronales Netz mit n_I Eingangs-, n_H Latenten (verdeckten) und n_0 Ausgangsneuronen verwendet. Die Eingangs-, Ziel und Gewichtskoeffizienten werden als Fuzzy-Zahlen dargestellt [19]. Die Vorgehensweise aus [54] wird mit Hilfe eines genetischen Algorithmus erweitert. Die Überlegungen für notwendige Vorgaben werden dadurch erheblich reduziert.

Es werden durch die Verwendung der α-Schnitte konventionelle scharfe (crisp) Lernregeln mit Hilfe von α-Schnitten für Fuzzy-Eingänge, gewünschte Ausgänge und Gewichte (connection weights) verwendbar

Die Neuronen der Eingangs-, Versteckten- und Ausgangsschichten werden im neuronalen Fuzzy-Netz jeweils wie folgt beschrieben:

$$O_{pi} = X_{pi},\ i = \overline{1, n_I} \tag{4.9}$$

$$O_{pj} = f(Net_{pj}),\ j = \overline{1, n_H} \tag{4.10}$$

$$Net_{pj} = \sum_{i=1}^{n_i} W_{ji} O_{pi} - \theta_j \tag{4.11}$$

$$O_{pk} = f(Net_{pk}),\ k = \overline{1, n_0} \tag{4.12}$$

$$Net_{pk} = \sum_{j=1}^{n_H} W_{kj} O_{pj} - \theta_k \tag{4.13}$$

Durch die Verwendung von α-Schnitten, werden die obigen Gleichungen in die folgende Form transformiert:

$$\left[O_{pi}\right]_\alpha = \left[X_{pi}\right]_\alpha \tag{4.14}$$

$$\left[O_{pj}\right]_\alpha = f\left(\left[Net_{pj}\right]_\alpha\right) \tag{4.15}$$

$$\left[Net_{pj}\right]_\alpha = \sum_{i=1}^{n_I} \left[W_{ji}\right]_\alpha \left[O_{pi}\right]_\alpha - \left[\theta_j\right]_\alpha \tag{4.16}$$

$$\left[O_{pk}\right]_\alpha = f\left(\left[Net_{pk}\right]_\alpha\right) \tag{4.17}$$

$$[Net_{pk}]_\alpha = \sum_{j=1}^{n_H} [W_{kj}]_\alpha [O_{pj}]_\alpha - [\theta_k]_\alpha \tag{4.18}$$

Für alle Fuzzy-Eingänge und Gewichte sind die α-Werte nicht negativ, aus diesem Grund kann man diese Gleichungen wie folgt umschreiben:

$$[O_{pi}]_\alpha = [[O_{pi}]_\alpha^L, [O_{pi}]_\alpha^U] = [[X_{pi}]_\alpha^L, [X_{pi}]_\alpha^U] \tag{4.19}$$

$$[O_{pj}]_\alpha = [[O_{pj}]_\alpha^L, [O_{pj}]_\alpha^U] = [f([Net_{pj}]_\alpha^L), f([Net_{pj}]_\alpha^U))] \tag{4.20}$$

$$[Net_{pj}]_\alpha^L = \sum_{\substack{i=1 \\ [w_{ji}]_\alpha^L \geq 0}}^{n_I} [W_{ji}]_\alpha^L [O_{pi}]_\alpha^L + \sum_{\substack{i=1 \\ [w_{ji}]_\alpha^L < 0}}^{n_I} [W_{ji}]_\alpha^L [O_{pi}]_\alpha^U - [\theta_j]_\alpha^L \tag{4.21}$$

$$[Net_{pj}]_\alpha^U = \sum_{\substack{i=1 \\ [w_{ji}]_\alpha^U \geq 0}}^{n_I} [w_{ji}]_\alpha^U [O_{pi}]_\alpha^U + \sum_{\substack{i=1 \\ [w_{ji}]_\alpha^U < 0}}^{n_I} [W_{ji}]_\alpha^U [O_{pi}]_\alpha^L - [\theta_j]_\alpha^U \tag{4.22}$$

$$[O_{pk}]_\alpha = [[O_{pk}]_\alpha^L, [O_{pk}]_\alpha^U] = [f([Net_{pk}]_\alpha^L), f([Net]_\alpha^U))] \tag{4.23}$$

$$[Net_{pk}]_\alpha^L = \sum_{\substack{j=1 \\ [w_{kj}]_\alpha^L \geq 0}}^{n_H} [W_{kj}]_\alpha^L [O_{pj}]_\alpha^L + \sum_{\substack{j=1 \\ [w_{kj}]_\alpha^L < 0}}^{n_H} [W_{kj}]_\alpha^L [O_{pj}]_\alpha^U - [\theta_k]_\alpha^L \tag{4.24}$$

$$[Net_{pk}]_\alpha^U = \sum_{\substack{j=1 \\ [w_{kj}]_\alpha^U \geq 0}}^{n_H} [W_{kj}]_\alpha^U [O_{pj}]_\alpha^U + \sum_{\substack{j=1 \\ [w_{kj}]_\alpha^L < 0}}^{n_H} [W_{kj}]_\alpha^U [O_{pj}]_\alpha^L - [\theta_k]_\alpha^U \; . \tag{4.25}$$

$T_p = (T_{p1}, T_{p2}, ..., T_{pn0})$ ist der gewünschte Ziel-Vektor. Man kann die Erfolgsfunktion des Fuzzy-Ausganges für ein α-Werte-Netz wie folgt bestimmen:

$$e_{pk\alpha} = e_{pk\alpha}^L + e_{pk\alpha}^U \tag{4.26}$$

Hierbei sind $e_{pk\alpha}^L$ und $e_{pk\alpha}^U$ die mittleren quadratischen Fehler für jeweils die unteren und oberen Grenzwerte des α-Werte-Netzes.

$$e_{pk\alpha}^L = \alpha([T_{pk}]_\alpha^L - [O_{pk}]_\alpha^L)^2 / 2 \tag{4.27}$$

$$e_{pk\alpha}^U = \alpha([T_{pk}]_\alpha^U - [O_{pk}]_\alpha^L)^2 / 2 \tag{4.28}$$

Die Erfolgsfunktion für alle Paare (X_p, T_p) lautet:

$$e_p = \sum_{\alpha} \sum_{k=1}^{n_0} e_{pk\alpha} \tag{4.29}$$

Die gelernten Fuzzy-Gewichte sind als Fuzzy-Zahlen in Trapezform darstellbar. Demzufolge werden wir für jeden Gewichtskoeffizienten vier Parameter feststellen:

$$W_{kj} = \left(W_{kj}^{(1)}, W_{kj}^{(2)}, W_{kj}^{(3)}, W_{kj}^{(4)}\right) \tag{4.30}$$

Mit Verwendung des Error-Back-Propagation-Algorithmus wird für jeden Parameter $W_{kj}^{(q)}$, $q = \overline{1,4}$ das folgende berechnet

$$\Delta W_{kj}^{(q)}(t+1) = -\eta \frac{\partial e_{p\alpha}}{\partial W_{kj}^{(1)}} + \zeta \Delta W_{kj}^{(q)}(t), \quad q = \overline{1,4} \tag{4.31}$$

Hierbei ist η die Lernrate, ζ eine Konstante und t ein Versuchsschritt. Ableitungen werden wie folgt durchgeführt:

$$\frac{\partial e_{p\alpha}}{\partial W_{kj}^{(1)}} = (1-\alpha) \frac{\partial e_{pk\alpha}^{L}}{\partial \left[W_{kj}\right]_{\alpha}^{L}}, \tag{4.32}$$

$$\frac{\partial e_{p\alpha}}{\partial W_{kj}^{(2)}} = \alpha \frac{\partial e_{pk\alpha}^{L}}{\partial \left[W_{kj}\right]_{\alpha}^{L}}, \tag{4.33}$$

$$\frac{\partial e_{p\alpha}}{\partial W_{kj}^{(3)}} = h \frac{\partial e_{pk\alpha}^{U}}{\partial \left[W_{kj}\right]_{\alpha}^{U}}, \tag{4.34}$$

$$\frac{\partial e_{p\alpha}}{W_{kj}^{(4)}} = (1-h) \frac{\partial e_{pk\alpha}^{U}}{\partial \left[W_{kj}\right]_{\alpha}^{U}}, \tag{4.35}$$

Vorausgesetzt, es gibt m Lernpaare (X_p, T_p), $p = \overline{1,m}$, wird der nachfolgende genetische Algorithmus aufgerufen, der die notwendigen α_i-Werte (d.h. α_1, α_2, …, α_n) bestimmt, es gilt dann der folgende Lernalgorithmus:

1. Schritt: Wiederholung der 2. bis 5. Schritte für $\alpha_i, i = \overline{1, n};$

2. Schritt: Wiederholung der 3. bis 5. Schritte für $p_i, i = \overline{1, m};$

3. Schritt: Berechnung des α-Werte-Netzes des Fuzzy-Ausgangsvektors O_p, der dem Eingangsvektor X_p entspricht;

4. Schritt: Bestimmung der vier Parameter der Fuzzy-Gewichte (Schwellenwerte) durch die Überprüfungsfunktion;

5. Schritt: Rückkehr zum 1. Schritt, falls die im Voraus gegebenen Stopbedingungen nicht erfüllt werden.

Als Stop-Kriterium kann man eine im Voraus angegebene Anzahl von Iterationen des Algorithmus verwenden.

Der verwendete genetische Algorithmus hat die folgende Struktur:

1. Festlegung des möglichen Bereiches für α.
   Wird kein Bereich festgelegt, ist der Bereich von 0 bis 1. Eine Begrenzung auf nur einen Teilbereich ermöglicht eine feinere Unterteilung der Berechnung bei gleicher Rechenzeit.

2. Festlegung der Anzahl der Codierungsbits für α.

3. Codierung von α.

4. Generierung der initialen Population.
   Dies geschieht entweder willkürlich oder auf der Basis verfügbarer Informationen, wie sie z.B. aus vorherigen Versuchen stammen können.

5. Verwendung der Population im 1. Schritt des Lernalgorithmus

6. Überprüfung der Werte mit der Fitnessfunktion e_p (Gl. 4.29)

7. Abbruch nach vorher bestimmter Anzahl von Durchläufen

4.4 Genetische Algorithmen zum Lernen neuronaler Fuzzy-Netze

Eine weitere Möglichkeit für das Lernen in neuronalen Fuzzy-Netzen stellen die Genetischen Algorithmen dar. Diese Methode wird notwendig, da die in 4.3 vorgestellt Methode nicht nur Vorteile hat. Auf Grund dessen, das sie u.a. auf der Gradientenmethode aufbaut, hat sie die gleichen Nachteile. So kann der ganze Lernalgorithmus zu einem lokalen Extremwert laufen. Ebenso ist die Lerngeschwindigkeit von der Genauigkeit abhängig. Ein Ausweg hieraus ist die Verwendung der Genetischen Algorithmen. Hierbei ist es nicht von Nöten, dass die Funktionen stetig sind. Außerdem ist es auf Grund des Mutationsoperators möglich auch unter Umständen den globalen Extremwert zu finden.

Nachfolgend wird nun ein 3-schichtiges Fuzzy feed-forward Netzwerk des Typs 3 betrachtet [19, 24]. Man kann aber auch jede beliebige Anzahl von Lagen verwenden. Die Eingangs-Neuronen werden in Form eines Vektors von Fuzzy-Eingangssignalen dargestellt.

$$X = \{x_1, x_2, ..., x_n\}$$

Die Eingangs-Neuronen führen an den Eingangssignalen keine Operationen aus, sie übertragen sie einfach an alle Fuzzy-Neuronen der nachfolgenden (versteckten) Schicht. Die Verbindungen zwischen den Schichten werden als Matrix von Fuzzy-Gewichten W^h dargestellt. Das Element w_{ij}^h ist ein Fuzzy-Gewicht zwischen dem i-ten Neuron der Eingangsschicht und dem j-ten Neuron der versteckten Schicht. Der gesamte Fuzzy-Eingang des j-ten Neuronen der zweiten Schicht wird definiert durch:

$$I_j^h = \sum_{i=1}^{N} w_{ji}^h \cdot x_i - \theta_j^h \qquad (4.36)$$

Hierbei ist I_j^h der gesamte Fuzzy-Eingang des j-ten Neurons der versteckten Schicht, x_i ist der i-te Fuzzy-Eingang dieses Neurons, und θ_j^h ist die Fuzzy-Schwelle des j-ten Neuronen.

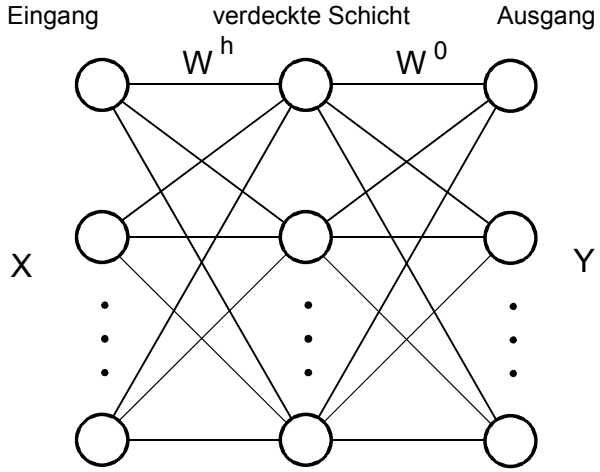


Bild 4.2: Dreischichtiges neuronales Fuzzy Netz

Der Fuzzy-Ausgang des j-ten Neurons ist durch die folgende Übertragungsfunktion definiert:

$$y_j^h = f(I_j^h). \tag{4.37}$$

Der Ausgang wird mittels (4.37) und des Erweiterungssprinzips von Zadeh [55] berechnet. Des Weiteren wird die folgende sigmoide Funktion verwendet:

$$f(x) = \frac{1}{1+e^{-x}} \ . \tag{4.38}$$

Diese Ausgänge werden dann an die Neuronen der folgenden Schicht weitergegeben, wo dann die gleichen Operationen durchgeführt werden. In einem dreischichtigen neuronalen Netz ist die folgende Schicht die Ausgangsschicht. Hier wird der Ausgangsvektor Y erzeugt. In allgemeinen mehrschichtigen Neuronalen Fuzzy Netzen breiten sich die Fuzzy-Signale Schicht für Schicht bis zur Ausgangsschicht aus.

Das Trainieren von neuronalen Fuzzy Netzen ist das Entwerfen eines neuronales Fuzzy Netzes, welches ein erwünschte Verhalten zeigt, d.h. es erzeugt erwünschte Fuzzy-Ausgänge bei einer Anregung an den Fuzzy-Eingängen. Eine Paarmenge von gegebenen Fuzzy-Eingängen/erwünschten Fuzzy-Ausgängen stellt eine Trainingsmenge für das neuronale Fuzzy Netz dar:

$$\left(X_i; Y_i^*\right), \quad i = \overline{1,N},$$

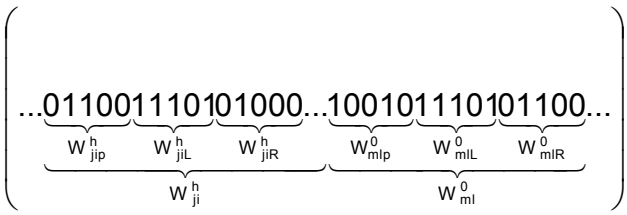
Hierbei ist N die Zahl der Paare in der Trainingsmenge. Der Lernalgorithmus ermittelt Fuzzy-Werte für die Gewichte zwischen den einzelnen Neuronen so, dass die Fuzzy-Distanzen zwischen dem berechneten und dem erwünschten Ausgang für alle Fuzzy-Eingangsvektoren in der Trainingsmenge minimiert wird. Dies bedeutet:

$$d = \sum_{i=1}^{N} \left| Y_i^* - Y_i \right| \rightarrow \min . \tag{4.39}$$

Hier ist Y_i^* der erwünschte Ausgang für das i-te Trainingspaar und Y_i der tatsächliche Ausgang des neuronalen Fuzzy Netzes. Um das Ziel (4.39) mit traditionellen Methoden zu erreichen (z.B. gradientenbasierende Techniken), muss man α-Schnitte einführen (vgl. Kap 4.3), da die Gleichung 4..39 nicht für sich allein differenzierbar ist.
Zum Trainieren von neuronalen Fuzzy Netzen benutzt man das Distanzmaß (4.39) als Basis für eine Fitnessfunktion. Man definiert die Fitnessfunktion wie folgt:

$$F = \frac{1}{(1+d)} . \tag{4.40}$$

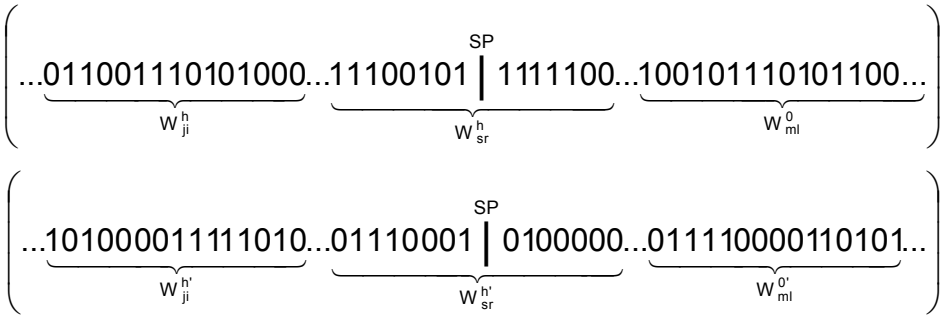
Je größer die Distanz zwischen dem tatsächlichen Ausgang des neuronales Fuzzy Netz und dem erwünschten Ausgang ist, umso kleiner ist der Fitnesswert und umgekehrt.
Die Repräsentation von möglichen Lösungen ist eine weitere wichtige Frage in der durch Genetische Algorithmen basierten Optimierung. Traditionell benutzen Genetische Algorithmen eine binäre Entschlüsselung, um Chromosomen zu erzeugen. Es wird also eine angemessene Abbildung von Problemvariablen zu binären Strings, und umgekehrt, verlangt. In diesem Fall trägt ein einzelnes Chromosom den Wert von allen Fuzzy-Gewichten des neuronalen Fuzzy Netzes in einem binären Format. Die benutzen Gewichte, werden als LR-artige Fuzzy-Zahlen mit n Parametern ausgedrückt. Ein Fuzzy-Gewicht wird durch n binäre Strings einer spezifizierten Länge dargestellt. Jedes der n binären Strings korrespondiert mit einem bestimmten Parameter eines Fuzzy-Gewichtes.
Das folgende Chromosom zeigt die dreieckige Darstellung von Fuzzy-Gewichten. Hier wird die Genauigkeit von fünf Bit pro Parameter eines Fuzzy-Gewichtes verwendet:



Hier ist w^h_{ji} ein Fuzzy-Gewicht zwischen dem j-ten versteckten und dem i-ten Eingangsneuron, $w^h_{jip}, w^h_{jiL}, w^h_{jiR}$ sind die Spitze (peak), die linke und die rechte Ausdehnung eines Fuzzy-Gewichtes. Ähnlich ist w^0_{ml} ein Fuzzy-Gewicht zwischen dem m-ten Ausgang und dem l-ten versteckten Neuron.

Die Strings werden in reale Zahlenwerte für die Gewichte umgewandelt. Es existiert also für jede Kombination von Nullen und Einsen ein neuronales Fuzzy Netz mit den entsprechenden Gewichten. Hiefür wird dann der Abstand berechnet und mit Hilfe von (4.40) in einen Fitnesswert für ein gegebenes Chromosom umgewandelt.
Der Selektionsoperator wählt willkürlich Chromosomen aus der Population aus, um eine neue Population zu produzieren.
Die genetischen Operationen, Kreuzung und Mutation, werden auf die neue Population angewandt. Der Kreuzungsoperator wählt willkürlich zwei Chromosomen, kreuzt sie und produziert somit zwei Nachkommen-Chromosomen. Hierfür wird der Kreuzungspunktpunkt in den Chromosomen willkürlich gewählt. Dann werden Nachkommen-Chromosomen generiert, die alle die Gene eines Elternteils bis zum Kreuzungspunkt und von diesem Punkt an die Gene des anderen Elternteils erben.

Als Beispiel [19] dienen die folgenden zwei Chromosomen, die zwei verschiedene Gruppen von möglichen dreieckigen Gewichten für ein neuronales Fuzzy Netz darstellen.



Hier steht SP für „split point“ (Kreuzungspunkt). Nach der Kreuzung werden die Nachkommen-Chromosomen zu:

$$\left(\ldots \underbrace{0110011101010 00}_{W^{h}_{ji}} \ldots \underbrace{11100101 \overset{SP}{|} 0100000}_{W^{h''}_{sr}} \ldots \underbrace{0111100001101 01}_{W^{0'}_{ml}} \ldots \right)$$

$$\left(\ldots \underbrace{1010000111110 10}_{W^{h'}_{ji}} \ldots \underbrace{01110001 \overset{SP}{|} 1111100}_{W^{h'''}_{sr}} \ldots \underbrace{1001011101011 00}_{W^{0}_{ml}} \ldots \right)$$

Es ist sehr wahrscheinlich, dass eines der Gewichte keinem der Eltern gehört, da der Spaltpunkt innerhalb des Sub-Strings fällt, der eines der Fuzzy-Gewichte darstellt.
Die Mutation wählt willkürlich eines der Bits in einem Chromosom und ändert es.

Das folgende Beispiel zeigt den Mutationsoperator:

$$\left(\ldots \underbrace{0110011101010 00}_{W^{h}_{ji}} \ldots \underbrace{1110010 \overset{Mt}{1}1111100}_{W^{h}_{sr}} \ldots \underbrace{1001011101011 00}_{W^{0}_{ml}} \ldots \right)$$

Nach der Mutation ergibt sich folgendes Bild:

$$\left(\ldots \underbrace{0110011101010 00}_{W^{h}_{ji}} \ldots \underbrace{1110011 \overset{Mt}{1}1111100}_{W^{h'}_{sr}} \ldots \underbrace{1001011101011 00}_{W^{0}_{ml}} \ldots \right)$$

Hier deutet Mt auf das mutierte Gen. Tatsächlich verändert die Mutation willkürlich eines der Fuzzy-Gewichte eines neuronalen Fuzzy Netzes, welches vom Chromosom dargestellt wird. Der Algorithmus geht wie folgt vor [19]:

1. Ursprüngliche Population wird willkürlich erschaffen (d.h. eine Population von Fuzzy-Gewichten vom einem neuronalen Fuzzy Netz wird willkürlich festgelegt).
2. Für jedes Chromosom wird sein Fitnesswert berechnet.
3. Eine Selektion findet statt, um eine neue Population zu schaffen.

4. Der Kreuzungsoperator wird auf die Population angewandt.
5. Der Mutationsoperator wird auf die Population angewandt.
6. Sind die Stoppkriterien nicht erreicht, dann weiter zu Schritt 2.
7. Ende.

Das Stoppkriterium könnte zum Beispiel die maximale Anzahl der Generationen sein.

Weiterhin hat die Erfahrung mit diesem Problems gezeigt, dass die α-Schnitt-Methode mit erheblicher rechnerischer Komplexität verbunden ist.

5 Struktur und Realisierung

Dieses Kapitel beschreibt den grundsätzlichen Aufbau eines Systems zu Fahrzustandbestimmung. Hierfür wird aus einer Vielzahl von Eingangswerten, wie verschiedene Temperaturen und Geschwindigkeiten unter der Verwendung verschiedener Untersysteme ein Gesamtbild des Fahrzustandes ermittelt. Zu den Untersystemen gehören unter anderem ein System zur Überprüfung von Eingangswerten und ein System zur Bestimmung des Fahrerverhaltens. Bild 5.1 zeigt den allgemeinen Aufbau eines solchen Systems. In Bild 5.2 ist der schematischen Aufbau des gesamten Systems dargestellt [10-16].

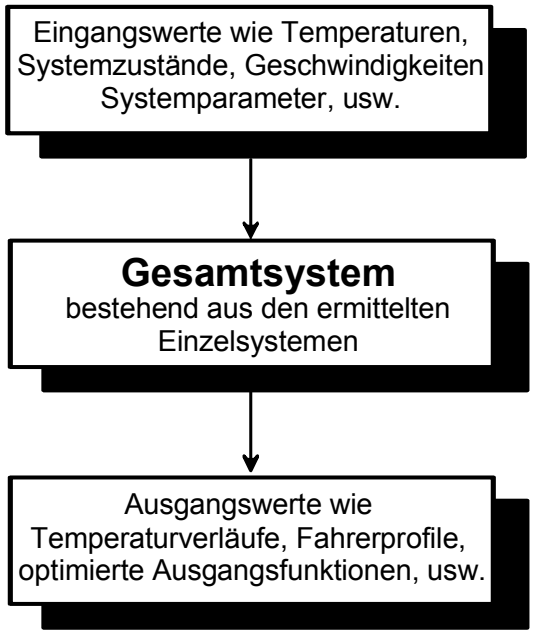


Bild 5.1: Grundsätzlicher Aufbau

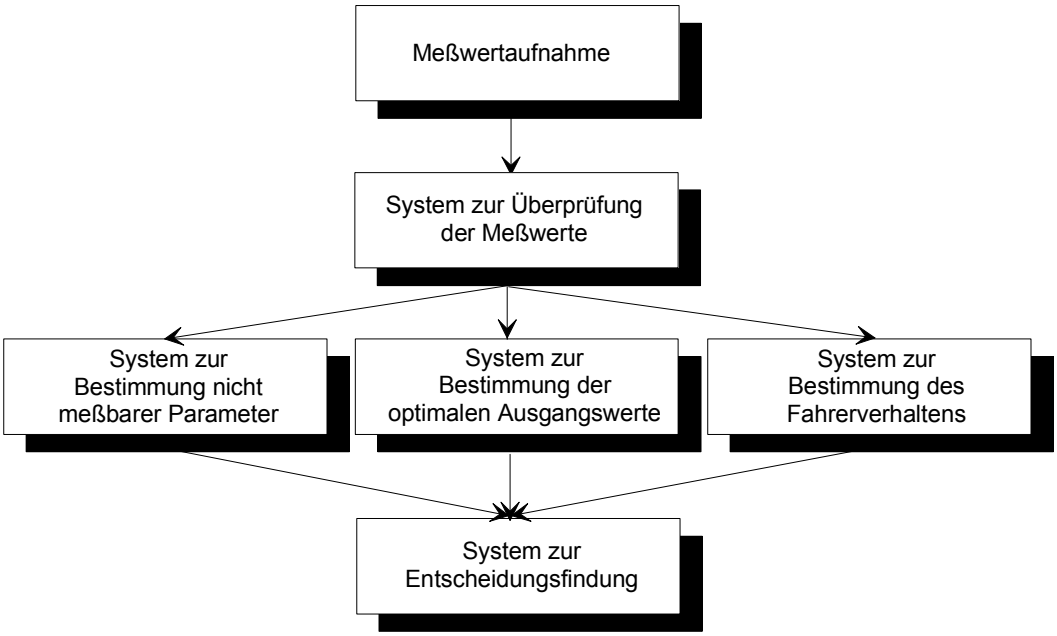


Bild 5.2: Schematischer Aufbau des Systems

Das gesamte System besteht aus autonomen Einzelsystemen. Dieser modulare Aufbau hat den Vorteil, dass man Systemteile von Fall zu Fall aktivieren bzw. deaktivieren kann. Dies wiederum erlaubt eine optimale Anpassung an die jeweiligen Aufgabenstellungen. Es ist nicht immer notwendig, alle Verfahren zu verwenden. Es kann allerdings vorkommen, dass einzelne Modelle sowohl eine als auch mehrere Methoden des Soft-Computing beinhalten. Nachfolgend werden einzelne Systeme ausführlicher beschrieben.

5.1 Überprüfung der Eingangswerte

Dieses System dient zur Überprüfung der aufgenommenen Eingangswerte. Für eine Vielzahl von Messwerten muss zunächst geklärt werden inwieweit sie für die weitere Verarbeitung überhaupt verwendet werden können. Hierfür gibt es eine Vielzahl von linguistischen Beschreibungen wie zum Beispiel in [19] beschrieben. Ebenso sind bei den hier gemachten Untersuchungen noch eine Vielzahl von linguistischen Beschreibungen hinzugekommen. Als Beispiel können hier Werte angeführt werden, bei denen die Plausibilität bzw. Nicht- Plausibilität offensichtlich sind. Ein Temperatursturz einer Fahrbahntemperatur von 10°C in 5 Minuten ist eher unwahrscheinlich da das Temperaturverhalten sehr träge ist. Ein weiteres Ziel dieses Systems ist es, soweit möglich, fehlerhafte Messwerte zu erkennen und zu korrigieren. Dies ist u. U. nicht immer oder nur bedingt möglich. Sollten diese Methoden bei einem Messwert nicht das gewünschte Ergebnis liefern kann es von Vorteil sein diesen Messwert durch ein anderes System ermitteln zu lassen. Das System hierfür ist das System zur Ermittlung nicht direkt messbarer Parameter auf das später noch genauer eingegangen wird. Das System für die Überprüfung der Messwerte ist ein reines Fuzzy-Logik-System mit einer Wissensbasis aus linguistischen Regeln die aus Erfahrungen, einer Vielzahl von Tests und physikalischen Gesetzmäßigkeiten hergeleitet sind.

Das System besteht in diesem Fall aus einer Vielzahl linguistischer Variablen. Sollten bei der Verwendung des Systems noch zusätzliche Zusammenhänge oder mögliche Fehler auftreten, ist keine Neuprogrammierung notwendig, eine einfache Erweiterung der Wissensbasis und der Datei der linguistischen Variablen reicht. Die Wissensbasis verwendet IF- Then- Regeln. Der Grund hierfür ist die sehr gute Aussagekraft und die Möglichkeit, viele auftretende Phänomene so zu beschreiben. Als Defuzzyfizierung wird die Defuzzy-Methode für Singeltons gewählt. Es kommt hier nicht darauf an, das genaue Ergebnis zu ermitteln. Für diesen Fall reicht eine schnelle Näherungslösung. Es wird in diesem System nur entschieden ob der Messwert weiter verwendet werden kann oder ob er verworfen werden muss. Sollte letzteres der Fall sein, wird der Wert als fehlerhaft deklariert und das System kann geeignete Reaktionen starten. Hierzu zählen der Versuch den Wert erneut zu messen ebenso wie die Bestimmung des Wertes über das System zur Bestimmung bzw. Ermittlung nicht direkt messbarer Parameter. Als Beispiel für diesen Wert kann wiederum die Fahrbahntemperatur angesehen

werden. Sie ist abhängig von externen Größen wie die Temperatur der Luft, die Geschwindigkeit des Windes und einiger anderer Parameter und kann mit hinreichender Genauigkeit für einen bestimmten Bereich ermittelt werden. Bild 5.3 zeigt den Ablaufplan dieses Systems.

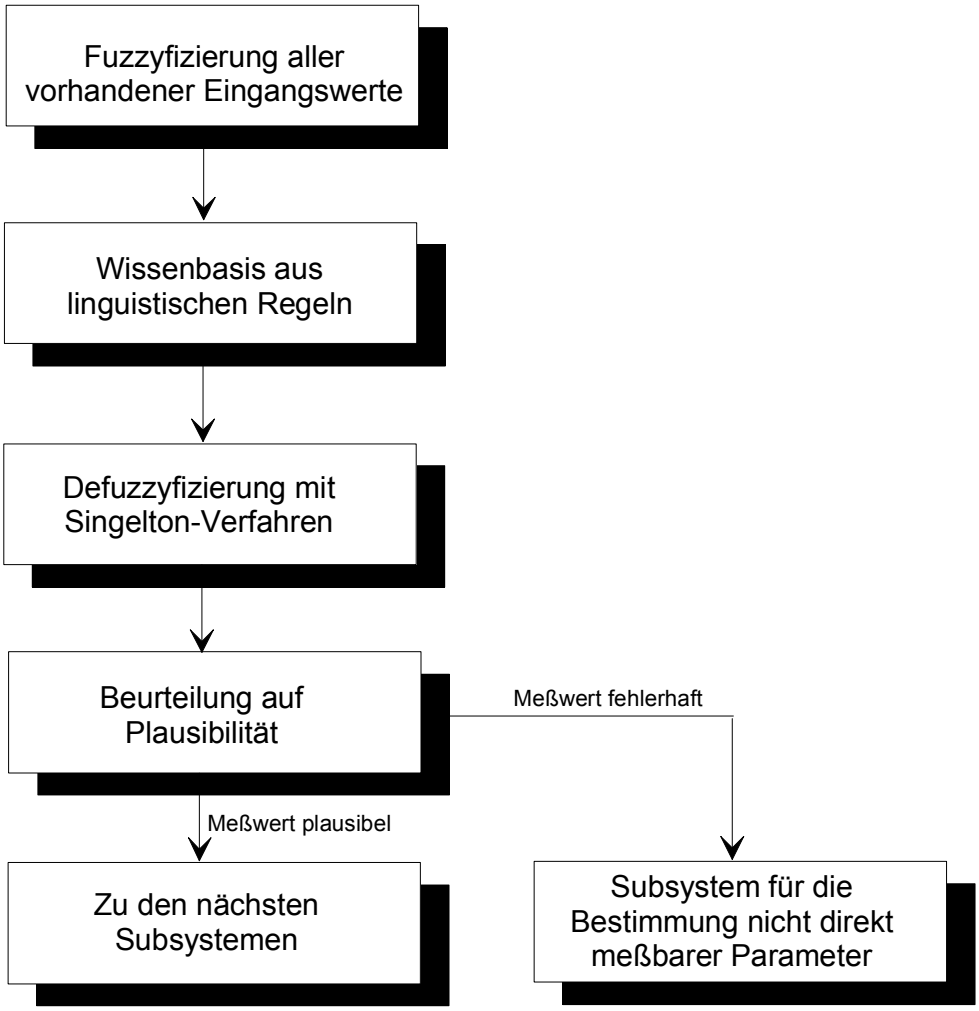


Bild 5.3: Ablaufplan für das System zur Überprüfung der Eingangswerte

Bei der hier beschriebenen Realisierung wurden die Eingangswerte für die Fahrbahntemperatur, die gemessenen Abgaswerte, die Restsalzmenge, die Lufttemperatur, die Luftfeuchte, die Taupunkttemperatur, die Pedalbedienungshäufigkeit und einige andere mehr überprüft.

Ein Teil der fehlerhaften Messwerte werden allerdings mit Hilfe einer Konfigurationsdatei schon herausgefiltert, da durch die dort angegebenen Messbereichsendwerte schon eine Einschränkung vorgenommen wurde. Sollte ein Wert außerhalb dieses Bereiches liegen, wird er direkt als fehlerhaft deklariert!

5.2 Bestimmung nicht direkt messbarer Messwerte

Es dient dazu, „Messwerte“ bzw. „Parameterwerte“ zu ermitteln. Es kann sich hierbei um verschiedene Arten von Messwerten handeln.

5.2.1 Nicht direkt messbare Messwerte

Diese Werte sind nicht direkt messbar, da entweder bauliche beziehungsweise systembedingte Einflüsse dies verhindern. Ein Beispiel hierfür ist die Temperatur der Nockenwelle eines Motors, die aufgrund des Einbaus im Motorblock und der großen Umlaufgeschwindigkeit nicht zum Beispiel durch einen PT100 Temperatursensor messbar ist, da dieser u.a. die Drehzahlen nicht aushalten würde. Ein intelligentes System kann allerdings mit Hilfe der Öltemperatur, der Umdrehungszahl, und der Zylindertemperatur näherungsweise die Temperatur der Welle bestimmen. Ebenso gehören hierzu die Messwerte, welche die Überprüfung auf ihren Messbereich nicht bestanden haben. Es ist daher unbedingt erforderlich, dass notwendige oder gar kritische Werte in diesem Modul nachgebildet werden. Wenn dies gemacht wurde, ist ein reibungsloser Betrieb auch in extremen Situationen, wenn auch eingeschränkt noch möglich. Es ist allerdings zweckmäßig, diese Option nur für eine bestimmte Zeit zu verwenden, da es auf eine längere Zeit zu größeren Abweichungen kommen kann.

5.2.2 Subjektive Messwerte

Diese Werte sind in der Regel Interpretationen. Beispiele hierfür sind Fahrzustandsbestimmungen welche an Hand von Fahrzeugendaten ermittelt werden [46] oder z.B. die Glätteprognose. In der Regel werden die Interpretationen in Form von linguistischen Regeln oder auch in Form von Tabellen linguistischer Regeln (TLR) beschrieben.

5.3 Fahrzustandsbestimmung

Nachfolgend ist ein System für eine Fahrzustandsbestimmung beschrieben:

Bild 5.4 zeigt den prinzipiellen Aufbau des Moduls zur Fahrzustandsbestimmung. Aufgrund seiner Struktur ist dieses System ein System mit reiner Fuzzy-Logik da es hierbei um die Bestimmung von Messwerten geht, deren Beschreibung existiert bzw. gut zu ermitteln ist.

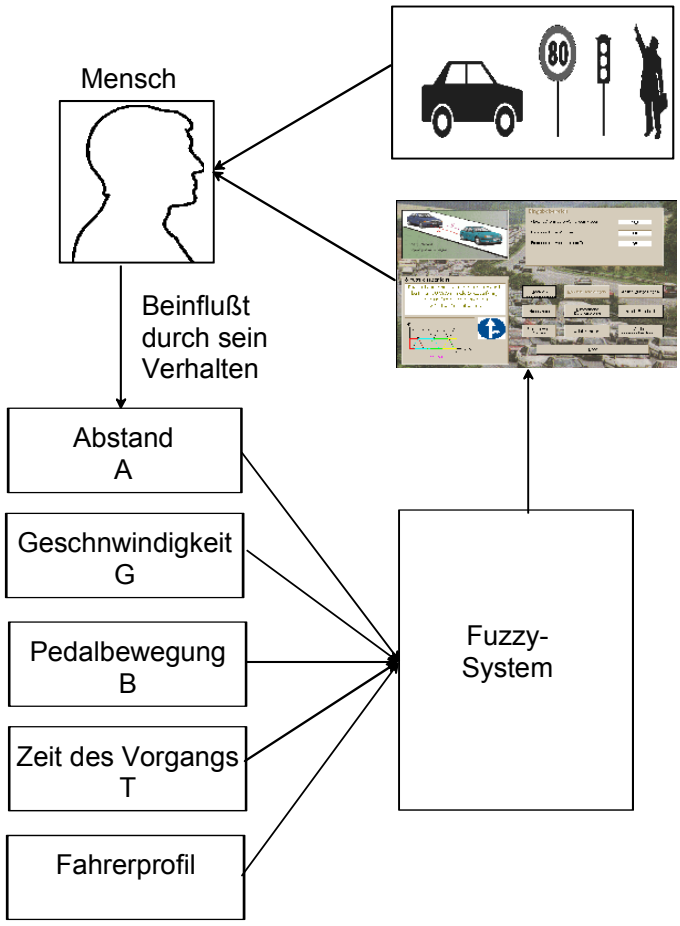


Bild 5.4: Modul zur Fahrzustandsbestimmung

Die Eingangsgrößen für das System sind:

a: Der Abstand zum vorausfahrenden Fahrzeug (A) in [m]
b: Die augenblickliche Geschwindigkeit (G) in [km/h]
c: Die Bremspedalbewegung pro Bremsvorgang (B) in [Grad]
d: Die Zeit des Bremsvorgangs (T) in [s]
e: Das Fahrerprofil

Die Ausgangsgröße für das System ist:

f: Die aktuelle Staugefahr (S) in [%]

Anschaulich ist dies mit einem Beispiel darzustellen:

Der Fahrer des Wagens reagiert aufgrund eines Bremsmanövers des Vordermannes bei einer Geschwindigkeit von 80 km/h des eigenen Fahrzeugs und bei einem Abstand von 100 Metern zum vorausfahrenden Fahrzeug mit einer Bremspedalbewegung von 30 Grad für 1,2 Sekunden.

G = 80 km/h A = 100 m B = 30 ° T = 1,2 s

Dies würde eine Stauwahrscheinlichkeit von 50% bedeuten, da es sich entweder nur um ein Abbremsen aufgrund eines plötzlichen Fahrmanövers handeln kann oder aber unter Umständen um den Beginn eines Staus, den man noch nicht sieht.

In Bild 5.5 ist der Bearbeitungsbildschirm dargestellt. Er ist so ausgelegt, dass die wichtigsten Werte klar erkennbar sind. Dies ist für dieses System notwendig, da die Regeln durch Versuche ermittelt werden. Die Messwertaufnahme ist relativ einfach, das System wird über einen Laptop an ein Versuchsfahrzeug angeschlossen. Wenn ein Bremsmanöver erfolgt ist der Zustand durch den Fahrer zu charakterisieren. Es kann entweder:

- Stau
- zähfließender Verkehr
- ein Hindernis auf der Fahrbahn
- freie Fahrbahn
- Unachtsamkeit

vorliegen.

Auf Grund der, von dem Fahrer gemachten Eingaben, wird die Zugehörigkeitsfunktionen ermittelt. Dies geschieht über eine statistische Auswertung der aufgenommenen Daten. Es ist allerdings anzumerken, dass es sich hierbei um subjektive Eingaben handelt. Ein anderer Fahrer reagiert auch anders. Durch die Erweiterung auf verschiedene Fahrer ist eine Verallgemeinerung möglich. Ein

besseres Ergebnis erzielt man aber, indem man das Fahrerverhalten mit berücksichtigt. Der Ansatz hierfür wurde in Kapitel 5.4 behandelt.

In Bild 5.5 ist der Hauptbildschirm der Fahrzustandsbestimmung zu sehen

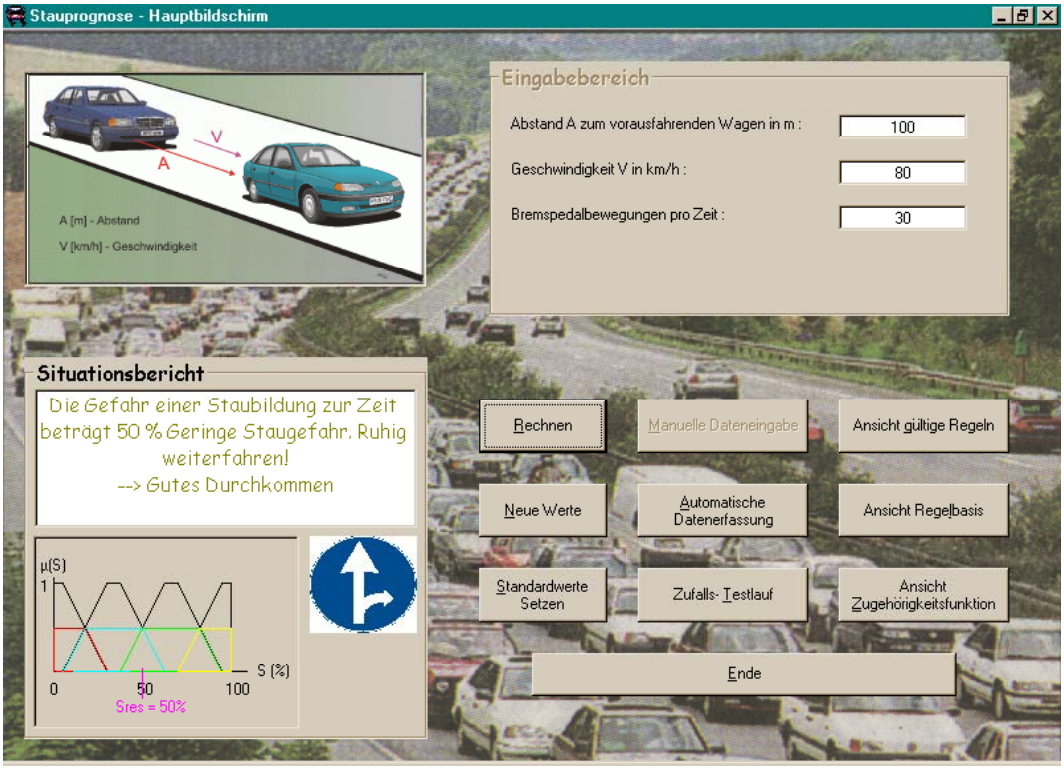


Bild 5.5: Hauptbildschirm

Um eine bessere Anpassung zu ermöglichen, sind auch bei diesem System die aktiven Regeln anzeigbar. Dies ist in Bild 5.6 dargestellt.

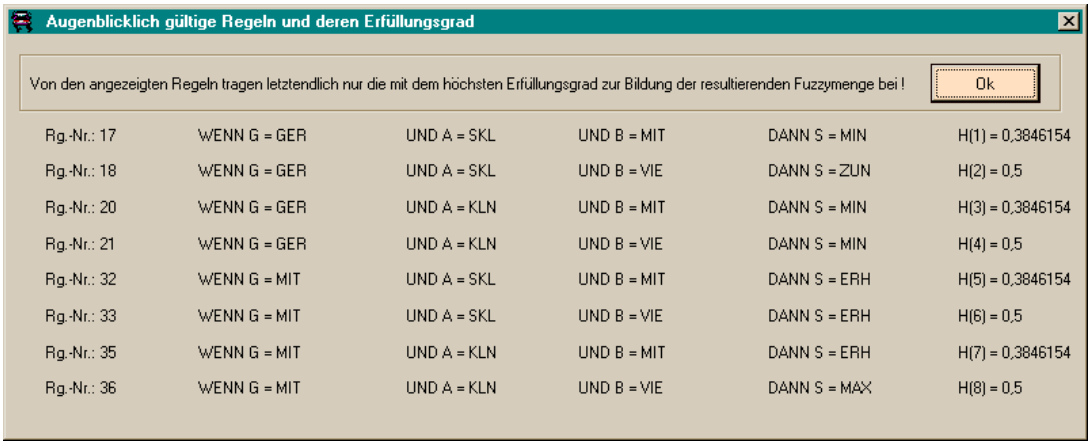


Bild 5.6: Ansicht der gültigen Regeln

Ein Auszug der verwendeten linguistischen Regeln ist in Bild 5.7 dargestellt:

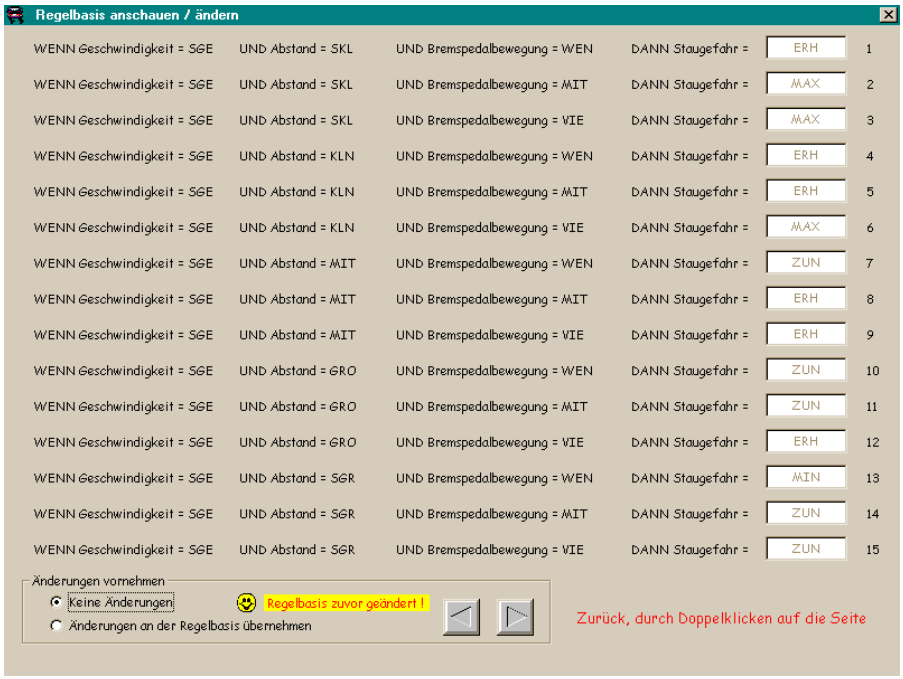


Bild 5.7: Regelbasis

Die Regeln in diesem Programmteil werden mit Hilfe eines speziellen Editors verändert. Anstelle des Editors kann auch ein adaptierendes System eingesetzt werden. Es bringt hier aber keinen Vorteil, da die Möglichkeiten zu Adaption im Fahrerverhalten liegen. Dies wird aber schon bei der Generierung des Fahrerprofils berücksichtigt Der verwendete Editor ist in Bild 5.8 dargestellt [46]

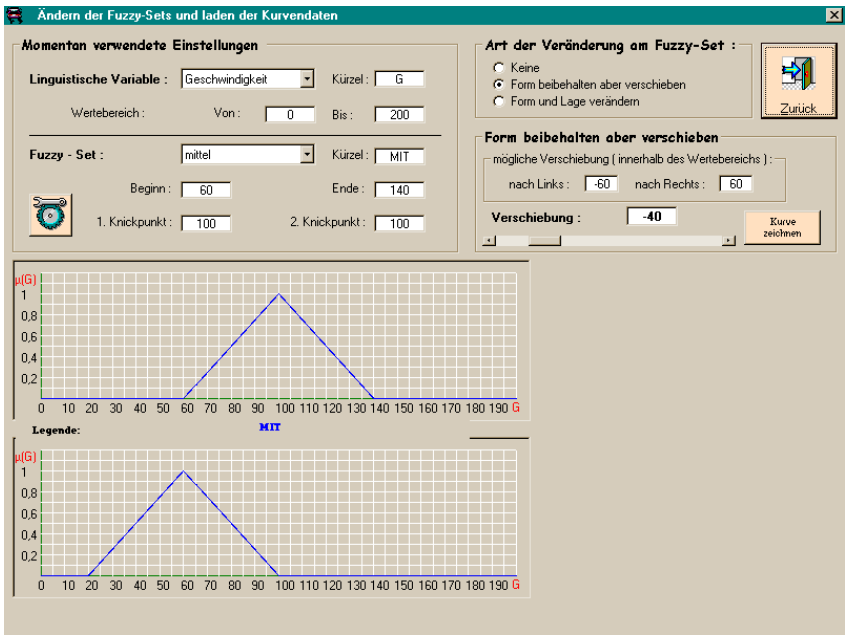


Bild 5.8: Editor für die linguistischen Variablen

Dieses System ermöglicht es mit begrenztem Aufwand, ein System zur Stauwarnung zu bauen, das seine Informationen einzig aus fahrzeugspezifischen Daten bezieht. Diese Daten stammen entweder von dem Fahrzeug selbst oder aus seiner nächsten Umgebung.

Durch zusätzlich von außen zugeführte Informationen über die Verkehrssituation in einigen Kilometern Entfernung, ist es dem System im Fahrzeug dann möglich den Fahrer viel früher auf eine bevorstehende Stausituation aufmerksam zu machen.

Der zweite Aspekt dieses Systems ist aber, einer intelligenten Motorsteuerung Informationen zur Verfügung zu stellen, aus denen erkennbar ist, wie wohl die nächsten Reaktionen des Fahrers sein werden. Diese Informationen sind wichtig, da sie einen Blick auf das Fahrverhalten des aktuellen Fahrers erlauben und eine optimale Motorsteuerung ermöglichen, welche den Benzinverbrauch und die Abgaswerte senken kann.

5.4 Erkennung des Fahrerverhaltens

Dieses System ermittelt durch einen Parallelbetrieb zum eigentlichen Prozess ein Bedienerprofil, welches die Reaktionen und Verhaltensweisen des Bedieners voraussehbar machen soll.

Dadurch lassen sich einige Reaktionen bereits im Voraus einleiten. Hierdurch werden wesentliche Optimierungen möglich, da man nicht nur kurzfristig planen kann, sondern auf einem längeren Bereich die Optimierungen ansetzt, was wiederum eine Verbesserung für die Lebensdauer und den Energie- und Rohstoffverbrauch bringt.

Aufgrund der Dynamik ändern sich die Prozesszustände und hierdurch auch die Reaktionen sehr schnell. Aus diesem Grund kommt ein Neuro-Fuzzy-System mit einer Adaption auf der Basis Genetischer Algorithmen zum Einsatz. Außerdem wird zusätzlich ein System aufgebaut, welches das Fahrerprofil über ein Neuronales Netz ermittelt, das mit Hilfe von α-Schnitten lernt. Es dient zur Überprüfung der gefundenen Ausgangsfunktionen. Der Vorteil in der Verwendung von α-Schnitten liegt darin, dass das Lernen in Neuro-Fuzzy-System auf die Verwendung konventioneller (scharfer) Lernregeln für die Fuzzy-Eingänge, Ausgänge und Gewichte zurückgeführt wird.

Um die Einstellungen zu optimieren greift dieses System auf drei allgemeine Profile zurück. Diese Profile sind in der Implementierungsphase entstanden und entsprechen drei durchschnittlichen Profilen. Das neue ermittelte Profil wird mit den bisher ermittelten und den vorgegebenen Profilen verglichen, ob es Übereinstimmungen gibt. Sind die Abweichungen nur minimal wird ein bekanntes Profil verwendet, welches sich in der Vergangenheit schon bewährt hat. Um dies festzustellen, liegt oberhalb dieses Systems noch ein einfaches Fuzzy-System um das passende Profil auszuwählen. Bild 5.9 zeigt ein solches System

Es ist bei dieser Anwendung von Vorteil die Profile von sehr rasanten und sehr vorsichtigen Fahrern zu unterscheiden. Diese beiden Gruppen unterscheiden sich in erster Linie durch ihren Fahrstil. Einem vorsichtigen Fahrer käme nicht in den Sinn immer sofort wenn er fahren kann direkt das Gaspedal durchzutreten. Er würde eher abwarten und untertouriger fahren als ein rasanter Fahrer, der schnell

von der Stelle kommen will. Es ist ein System zu entwerfen, welches beiden Typen gerecht wird.

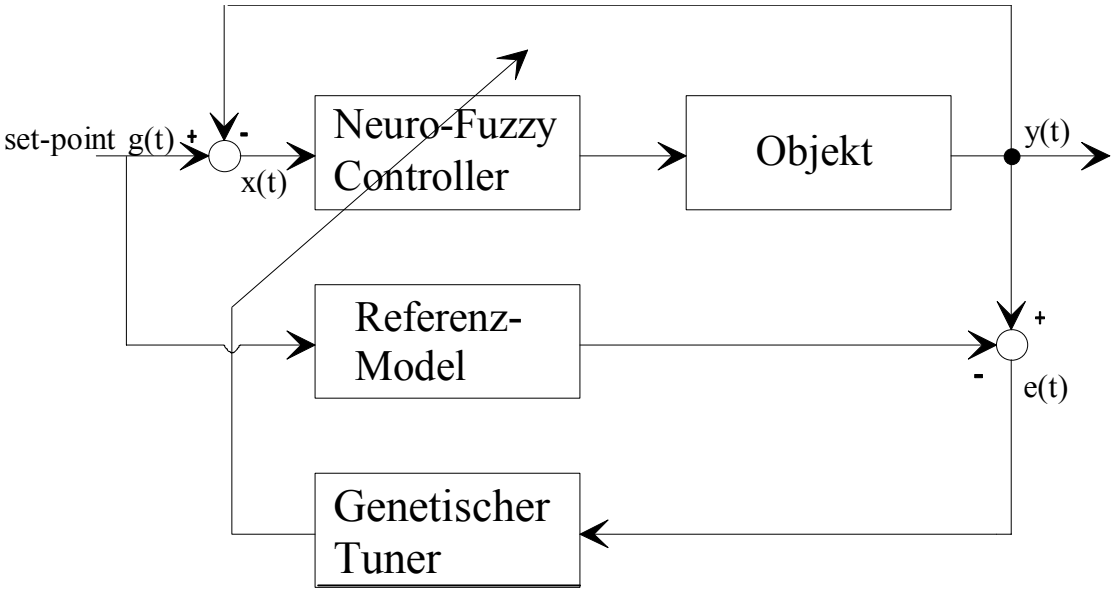


Bild 5.9: System für die Erkennung des Bedienerverhaltens

Aus diesen Gründen ist ein System entworfen worden, welches einen Ausweg aus diesen Problemen zeigt.

An das System wird eine Vielzahl von Anforderungen gestellt. Es muss in der Lage sein, nach einer vorgegebenen Trainingsphase Profile zu erzeugen, welche hier dem Fahrerverhalten entsprechen. Wie schon früher erwähnt, werden für die Realisierung zuerst drei Benutzerprofile erstellt. Es sind die Profile für vorsichtige, normale und aggressive Fahrer, es wären auch noch mehrere Profile vorstell- und realisierbar, es bringt aber keinen zusätzlichen Nutzen wie die Untersuchungen gezeigt haben.

Ein weiteres, wesentlich größeres Problem ist die Messwertaufnahme, die für dieses System notwendig ist, da es sich nur auf interne Werte verlassen kann. Ein solches System sollte autonom arbeiten und nicht auf manuelle Eingaben wie Stau oder ähnliches warten. Als Eingabemodul wurde das in Kapitel 5.2 dargestellte System verwendet, es ist in der Lage die notwendigen Daten aufzunehmen.

Die Eingangswerte des Systems sind:

- der Abstand zum vorausfahrenden Fahrzeug
  <sehr_gering, gering, mittel, groß, sehr_groß>

- die aktuelle Geschwindigkeit
  <langsam, mittel, hoch>

- die Bewegung des Bremspedals pro Zeit
  <wenig, mittel, viel>

- die Stauwahrscheinlichkeit
  <unwahrscheinlich, wahrscheinlich, sicher>

- die Glätteprognose (wenn vorhanden)
  <unwahrscheinlich, wahrscheinlich, sicher>

Die Ausgangswerte des Systems sind:

- Festlegung des Fahrerprofils
- <vorsichtig, durchschnittlich, aggressiv>

- Wahrscheinlichste nächste Reaktion
  <stark_bremsen, wenig_bremsen, gleichbleibend, wenig_gas, viel_gas>

Das in Kapitel 4.3 dargestellte Verfahren bietet wie Versuche gezeigt haben, hier leider nur eine langsame, aber nicht schlechtere Lösung. Aus diesem Grund wird ein System gemäß Kapitel 4.4, als reines Neuro-Fuzzy-Genetisches System aufgebaut [24].

Der Lernalgorithmus ist ein genetischer Algorithmus. Er codiert die jeweiligen Werte für die verwendeten Trapezfunktionen, für die Gewichte so, dass sie die Bedingungen von Kapitel 4.1 erfüllen. Dies ist notwendig um die Eindeutigkeit der Trapezfunktion zu erhalten.

Gemäß der, im vorherigen Kapitel beschriebenen Methoden, werden nun so viele Optimierungsdurchläufe generiert, bis die Ausgänge unter der vorgegebenen

zugelassenen Abweichung ε liegen bzw. bis eine vorher vorgegebene Anzahl von Optimierungsdurchläufen stattgefunden hat.

Da es hier nur um eine geringe Anzahl von Termen handelt, und die Zusammenhänge nicht im Vorhinein ganz klar sind, gibt es für Verknüpfungen und die beiden Ausgänge des Systems keine Vorgaben. Mann geht von einem voll beschriebenen System aus. Die Verknüpfungen die keine Vorteile bringen, bzw. unsinnig sind werden im Laufe des Lernprozesses erkannt. Dies ist dann offensichtlich, wenn die jeweiligen Gewichte gegen Null tendieren. Aus Vereinfachungsgründen werden alle Gewichte unter 0,05 auf Null gesetzt. Das Bild 5.10 [24] zeigt die Konfigurationsvorgaben für das neuronale Fuzzy Netz mit dem genetischen Lernalgorithmus:

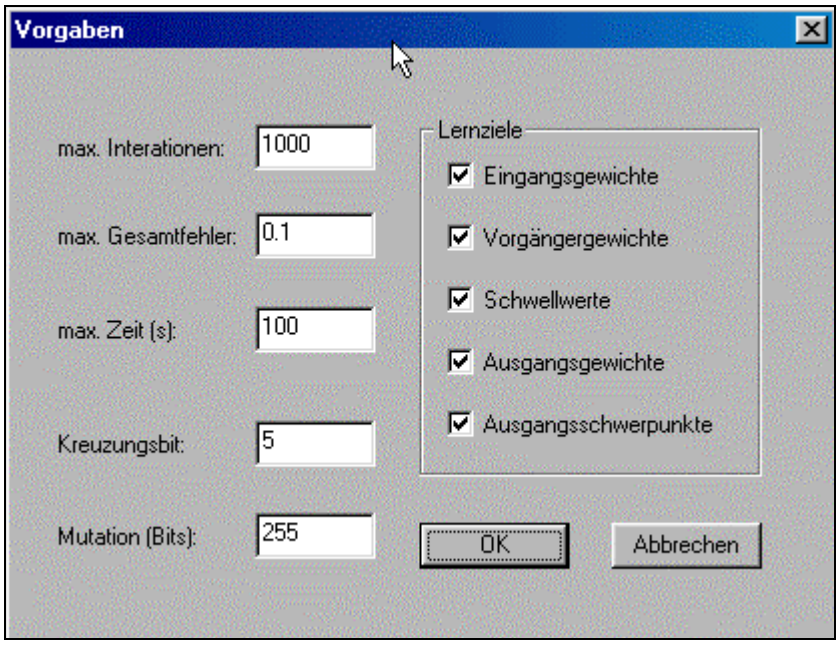


Bild 5.10: Konfigurationsvorgaben für das neuronale Fuzzy Netz mit dem genetischen Lernalgorithmus:

Die Ergebnisse der Optimierung für den Wert „Wahrscheinlichste nächste Reaktion“ sind stellvertretend für alle Werte in den Bildern 5.11, 5.12 und 5.13 dargestellt.

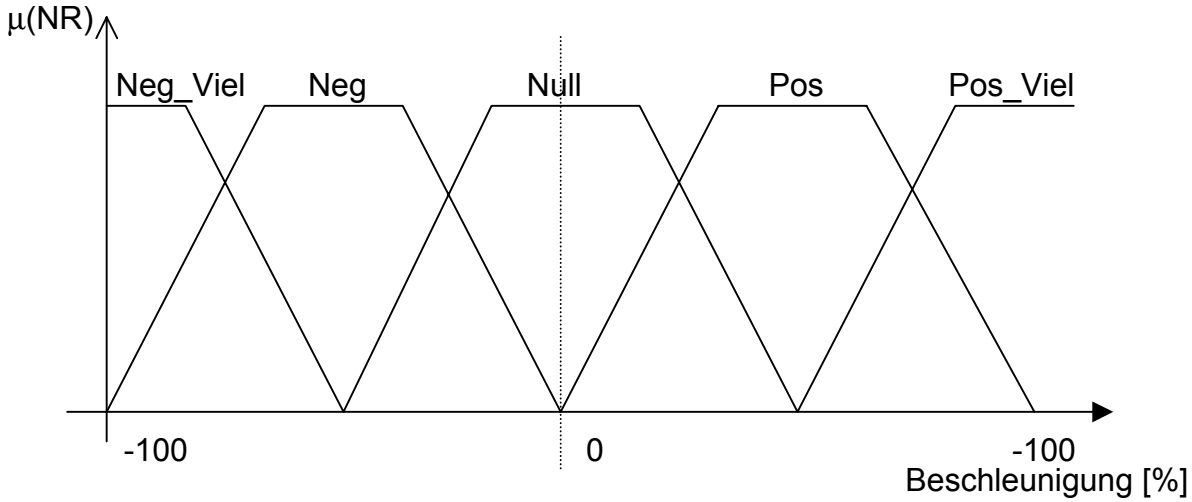


Bild 5.11: Zugehörigkeitsfunktion für „Wahrscheinlichste nächste Reaktion“ vor der Optimierung

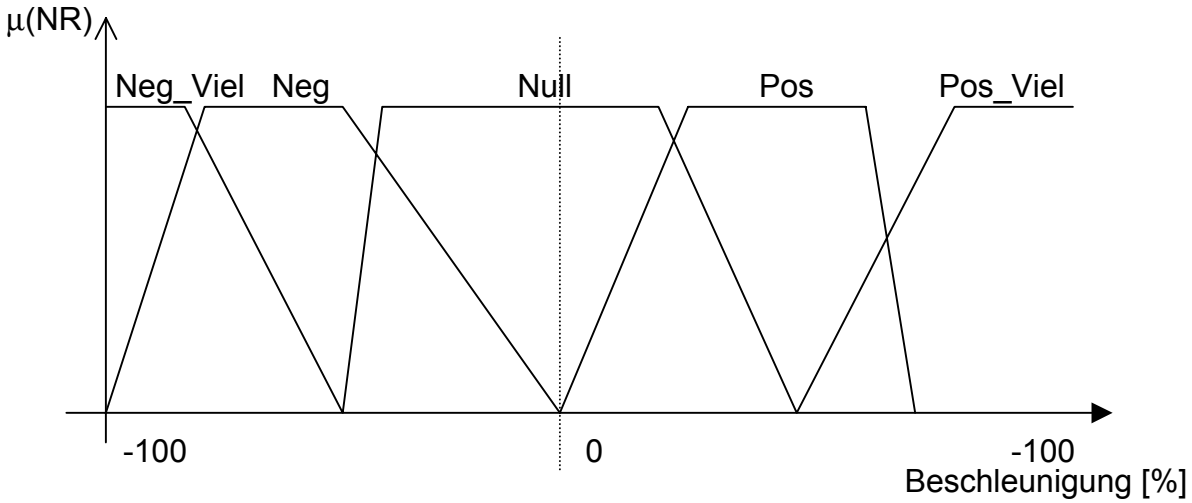


Bild 5.12: Zugehörigkeitsfunktion für „Wahrscheinlichste nächste Reaktion“ nach der Optimierung nach 4.3

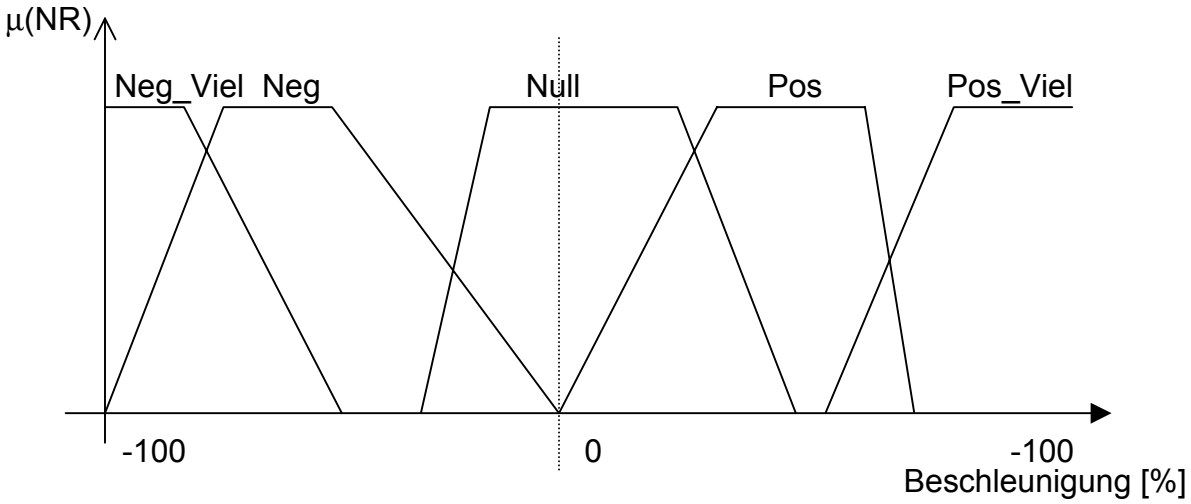


Bild 5.13: Zugehörigkeitsfunktion für „Wahrscheinlichste nächste Reaktion“ nach Neuro-Fuzzy- Genetischem Ansatz

Die Startwerte des Systems werden über einen Zufallsgenerator erzeugt. Als Einstellungen wurden die folgenden Werte verwendet:

Anzahl der Iterationen	= 1000
Max. Gesamtfehler	= 0,1
Max Zeit (s)	= 100
Kreuzungsbit	= 5
Mutation	= 500

Die Berechnungen durch das System [24] wurden mit Hilfe des Systems I3000 [47] der Firma Cosworth Technology und einer 80 Fahrzeuge umfassenden Studie überprüft. Es zeigt sich, dass das System das Fahrerverhalten in bestimmten Grenzen hinreichend genau ermitteln kann. Die Ausnahmen sind hier plötzlich auftretende Ereignisse, wie spielende Kinder usw.

5.5 Glätteprognose

Für diesen Systemteil hat die Firma Micks aus Oberstdorf die Testdaten zur Verfügung gestellt. Es wird mit Hilfe der Methoden der genetischen Algorithmen und durch die in Kapitel 4.2 beschriebene Methode ein Optimierungsdurchlauf generiert. Der genetische Optimierungsdurchlauf verwendet die Tatsache, dass praktisch nie ein vollständig beschriebenes System verwendet wird, indem er in die Lage versetzt wird die Wissensbasis zu erweitern. Man verwendet schon allein aufgrund der Bearbeitungszeit in den seltensten Fällen eine vollständige Verknüpfung sämtlicher zur Verfügung stehender Werte. Ebenso ist es in den meisten Fällen auch nicht möglich, sämtliche Zusammenhänge zu ermitteln. Aus diesen Gründen wird in diesem Bereich die Optimierung der Wissensbasis mit Hilfe von genetischen Algorithmen verwendet. Dies geschieht mit der Verwendung des genetischen Operators der Mutation um ein besseres Ergebnis zu erzielen.

Bild 5.14 zeigt den Aufbau des gesamten Systems.

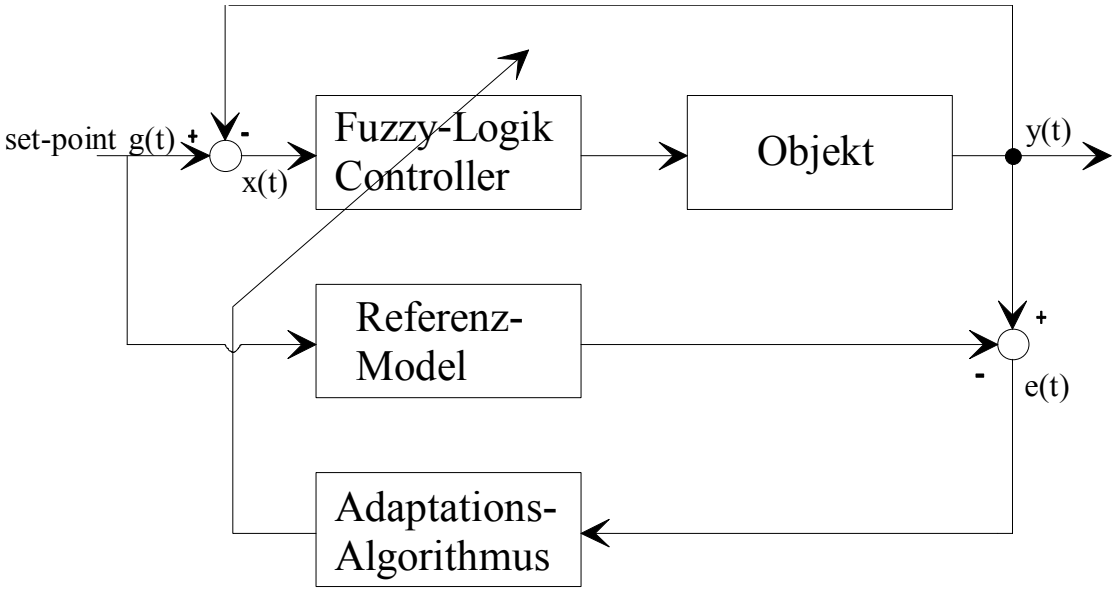


Bild 5.14: Allgemeiner Aufbau des Systems

Als Ausgangswert dieses Systems wird die Größe „Glättewahrscheinlichkeit“ ermittelt. Sie ist nicht physikalisch beschreibbar, sie dient mehr dazu, einen Anhaltspunkt zu geben, ob eine Glättesituation vorliegt. Eine hohe Glättewahrscheinlichkeit bedeutet, dass auch auf einer noch trockenen Straße eine Eissituation möglich ist, sobald es regnet.

Die Möglichkeit der Glätte durch Tau bzw. Kondensation wird hierbei durch eine Fuzzybeschreibung und Prognose des Taupunktes in Verbindung mit der Lufttemperatur und des Luftdruckes mit berücksichtigt. Plötzlich einsetzender Regen ist allerdings nicht auf eine gewisse Zeit im Voraus präzise vorherzusagen. Da in diesem Fall das Vorhersageintervall mindestens 90 Minuten betragen soll gibt die Glättewahrscheinlichkeit bei trockener Straße einen Richtwert an, wobei mit Eis zu rechnen ist, sobald es regnet. Vor Ort ist dies durch subjektive Beobachtungen möglich (dunkle Wolken usw.). Durch zahlreiche Versuche hat sich gezeigt, dass eine Glättewahrscheinlichkeit von 50% als Grenzwert zu nehmen ist, an dem es zu Eis- bzw. Reifglätte kommen kann. Die Gleichbehandlung dieser Zustände durch das System ist gewollt. Es ist unnötig diese zu trennen, da sich die Fuzzybeschreibungen gleichen, nur die Entstehung der zur Glätte notwendigen Nässe ist anders. Diese wird aber durch einen Regelzusatz in der Wissensbasis berücksichtigt.

Im Bild 5.15 wird der schematische Aufbau der Adaption gezeigt. Hierzu muss noch angemerkt werden, dass die Adaption erst etwa 90 Minuten später einsetzt, da hier erst die Prognose greift, und deswegen ja erst die Vergleichswerte der prognostizierten Ausgangswerte vorliegen. Es wurde zu Beginn zwar auch eine Adaption mit aktuellen Werten für die Glättewahrscheinlichkeit realisiert, dies erwies sich aber nicht als besonders vorteilhaft, da der aktuelle Bereich hinreichend gut allgemein beschrieben werden konnte und es deswegen reicht, ihn mit dem Prognoseteil zu optimieren.

Eine Adaption ist alleine schon aus dem Grund notwendig, dass man die jeweiligen Umgebungen nicht kennt oder sich diese verändern. Der Bereich der Glätte-prognose ist sowohl in einem Fahrzeug als auch als Insellösung für Fahrbahnen zu verwenden. Es soll aber nicht mit irgendwelchen unnötigen Konfigurationsdateien versucht werden, die Umgebung zu beschreiben, da die Adaption hinreichend genau arbeitet, wie die Überprüfung mit realen Werten (Bild 5.17) gezeigt hat.

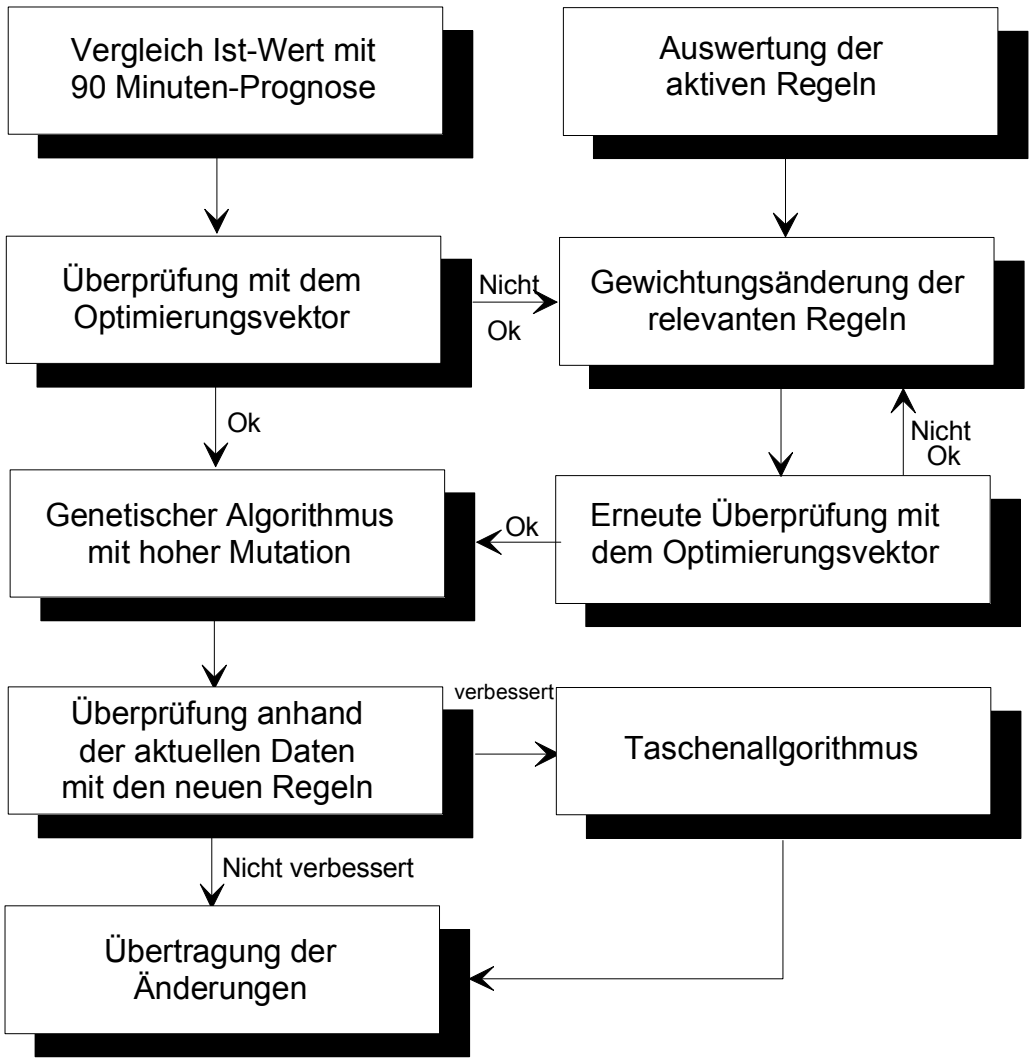


Bild 5.15: Schematischen Aufbau der Adaption.

Im Ist- Prognosewertvergleich werden die Prognosewerte für den Restsalzgehalt, den Gefrierpunkt, die Taupunkttemperaturdifferenz, der Fahrbahntemperatur, die Fahrbahnfeuchte und die Gefrierpunktdifferenz mit dem Istwert nach 90 Minuten verglichen.

Der einzige Punkt hierbei, der nicht immer mit Hilfe der Fuzzy-Logik beschrieben wird, ist der Wert für Restsalz. Das Problem mit dem Restsalz ist, dass der Wert nicht messbar ist, wenn die Straße trocken ist, obwohl Salz sich noch auf der Straße befindet. An dieser Stelle wird mit Hilfe der Gleichung (5.1) der Wert berechnet. Für diese Gleichung wird in den ersten Tagen ein so genanntes Abriebsmodell entwickelt. Sollte dann die Feuchtigkeit nicht mehr ausreichen bzw. gar nicht vorhanden sein, ist es trotzdem möglich eine relativ genaue Aussage zu bekommen, wieviel Restsalz noch auf der Straße liegt. Hierfür wird der Tag in

zwei Teile unterteilt: In den Zeitraum von sieben bis zwanzig Uhr und von zwanzig bis sieben Uhr. Der Grund hierfür liegt im Fahraufkommen auf der Straße. In dem ersten Bereich befindet sich eine wesentlich größere Anzahl von Fahrzeugen auf der Straße. Es wird nun in den Trockenphasen der Salzgehalt zu Beginn der Trockenheit gespeichert und der Wert für den Salzgehalt, wenn wieder genügend Feuchtigkeit zur Verfügung steht. Dieser Wert, wenn der Zeitraum nicht größer als drei Tage ist, wird durch die Anzahl der Stunden geteilt an denen Trockenheit herrschte. Es müssen hierbei allerdings die Tag- und Nachtstundenaufteilung berücksichtigt werden. Die ermittelten Werte werden nun zwischengespeichert. Wenn mindesten fünf Wertepaare gespeichert sind werden hiervon die Mittelwerte gebildet. Es werden auch im späteren Betrieb diese Mittelwerte immer nachgeführt. Dabei wird auf die letzten zehn Wertepaare zurückgegriffen. Die Restsalzprognose für die trockene Fahrbahn lautet wie folgt:

$$RS_i = Rs_{i0} - (RS_{MittelTagi} \cdot H_{Tag} + RS_{MittelNachti} \cdot H_{Nacht}) \qquad (5.1)$$

Hierbei ist: RS_i der Restsalzgehalt der i-ten Fahrbahn, Rs_{i0} der Restsalzgehalt der i-ten Fahrbahn zum Anfang der Trockenheit, $RS_{MittelTagi}$ die mittlere Restsalzgehaltabnahme der i-ten Fahrbahn am Tage, H_{Tag} die Stunden der Trockenheit am Tage, $RS_{MittelNachti}$ die mittlere Restsalzgehaltabnahme der i-ten Fahrbahn in der Nacht und H_{Nacht} die Stunden der Trockenheit in der Nacht.

Zu diesen Punkten parallel werden die aktiven Regeln der Wissensbasis ausgewertet. Es werden alle Regeln mit einem größeren Erfüllungsgrad als 0,1 als aktiv markiert. Die Regeln, deren Werte hierunter liegen, werden für die nachfolgenden Betrachtungen vernachlässigt.

Sind diese beiden Prozeduren abgeschlossen, wird das Ergebnis mit dem vorher angegebenen Optimierungsvektor verglichen. In diesem Fall sieht der Optimierungsvektor wie folgt aus.

OV = [Fahrbahntemperatur; Gefrierpunktdifferenz; Fahrbahnfeuchte; Taupunkt;
Glättewahrscheinlichkeit]

Die für die Optimierung verwendeten Werte sahen wie folgt aus:

OV = [0,2°C; 0,5°C; 20%; 0,2°C; 2%]

Steht nach der Überprüfung der Werte mit diesem Vektor fest, dass die Toleranzen eingehalten worden sind, geht es weiter zur genetischen Optimierung.

Wenn dies nicht der Fall ist, werden die Gewichtungsfaktoren der relevanten Regeln nach den Bedingungen von Kapitel 4.2 verändert. Danach wird das Gesamtsystem erneut mit allen Werten „gefüttert“ und das Ergebnis erneut überprüft. Dieser Durchlauf wird so lange durchgeführt, bis der gewünschte Vektor eingehalten wird. Dann geht es auch hier zur genetischen Optimierung.

Genetische Optimierung

Auf die grundsätzlichen Abläufe der genetischen Algorithmen wurde schon in Kapitel 3.4 eingegangen. Die Vorgehensweise bei dieser Optimierung ist hier mit kleineren Ausnahmen dieselbe.

Zuerst werden die einzelnen linguistischen Variablen in eine beliebige Reihenfolge gebracht, durchgezählt und die für die Beschreibung notwendige Bittiefe ermittelt. Dies geschieht über die nachfolgende Gleichung

$$N \geq \mathrm{ld}\, N_{LV} / \mathrm{ld}2 \qquad (5.2)$$

Hierbei steht N für die Bittiefe und N_{LV} für die Anzahl der linguistischen Variablen, der Wert, der hierbei berechnet wurde, wird auf den nächsten ganzzahligen aufgerundet.

Der Optimierungsstring ist 3N groß, da er drei Terme repräsentiert. In ihm sind zwei linguistische Eingangsvariablen und eine Ausgangsvariable codiert hinterlegt.

Für diese Anwendung ergibt sich demzufolge für die Eingangsseite:

$N \geq \mathrm{ld}\, 260 / \mathrm{ld}2$

$N \geq 8{,}1 \Rightarrow N = 9$ gewählt

und für die Ausgangsseite
$N \geq \mathrm{ld}\, 60 / \mathrm{ld}2$

$N \geq 5{,}9 \Rightarrow N = 6$ gewählt

Beispiel:

Der Wert für die „Fahrbahntemperatur_Null“ steht an der dritten Stelle und der Wert für den „Luftfeuchte_Wenig“ an der sechsten Stelle auf der Eingangsseite. Der Wert für „Eisglätte_sicher“ steht an der sechzigsten Stelle auf der Ausgangsseite.

Der String für die Regel sieht folgendermaßen aus (die Punkte dienen hier nur zur besseren Darstellung):

G_i = [000000011.000000110.111011]

Es wird nun die Population für den ersten Durchlauf gebildet. Sie besteht aus den abgebildeten Strings der bisherigen Wissensbasis und aus zufällig zusammengesetzten Größen. Es wird nun die initiale Population (IP) mit Hilfe der Fitnessfunktion überprüft.

In diesem Fall bedeutet die Fitnessfunktion keine reine mathematische Gleichung, sie ist viel mehr eine Rechenvorschrift:

- Systemüberprüfung mit realem Wertepaar, Markierung aller „aktiven“ Regeln (Strings) und Vergleich der Ergebnisse.

- Überprüfung der aktiven Regeln auf ihren Einfluss auf das Ergebnis (positiv/negativ) $\rightarrow$ Positiver Einfluss bedeutet, dass der sup x der Ausgangsgröße im sup x des tatsächlichen Wertes liegt. Wenn nicht ist die Bewertung negativ.

Danach werden die inaktiven Regeln (Strings) und die mit „negativ“ bewerteten Regeln aus der Population gelöscht.

Mit den Operatoren Kreuzung und Mutation gemäß Kapitel 3.4, werden aus den verbliebenen Strings neue Strings erzeugt bis die Population wieder vollständig ist.

Dieser Vorgang wird insgesamt n-mal, in dieser Arbeit n = 150, wiederholt und dann abgebrochen. Am Ende dieser Optimierung wird nun überprüft, ob neue

Regeln „überlebt“ haben. Dann wird eine angepasste Variante des „Taschenalgorithmus“ aufgerufen. Hierbei werden die neuen Regeln in eine „Tasche“ gepackt und weiter übernommen. Es ist nicht ratsam, diese Regeln direkt zu übernehmen, da es sich um einmalige Ergebnisse handeln kann. In dieser Tasche wird dann eine neue Wissensbasis generiert, welche die neuen Regeln mit einschließt. Diese neuen Regeln müssen bei mindestens zwei weiteren Durchläufen wieder vorkommen und „überleben“.

Ist auch das der Fall, muss sie sich in den nächsten Durchläufen bewähren. Ist das der Fall, wird sie endgültig in die neue Wissensbasis übernommen. Wenn das nicht der Fall ist, wird die neue Regel allerdings noch nicht fallen gelassen. Sie bleibt in der Tasche und wird in bestimmten Abständen wieder neu überprüft.

Für diese Anwendung haben sich die folgenden Einstellungen als vorteilhaft erwiesen und sind so realisiert worden.

Anfangsgröße der Wissensbasis : 61 Regeln

Kreuzungshäufigkeit : 3 Stellen
1.Stelle zwischen 3 und 6 Glied per Zufall
2.Stelle zwischen 12 und 16 Glied per Zufall
3.Stelle zwischen 19 und 22 Glied per Zufall

Mutationswert : 50 (jede 50. Stelle)

Populationsgröße : 120

Die hier verwendeten Eingangsgrößen und Fuzzy-Größen sind:

- Fahrbahntemperatur, Lufttemperatur, Bodentiefentemperatur1
  <sehr_kalt, kalt, null, warm, sehr_warm>

- Fahrbahnzustand
  <trocken, feucht, feucht/nass, nass, Schnee, Reif ,Glatteis, Eisglaette, Boschung>

- Restsalzgehalt, Luftfeuchte
  <wenig, mittel, viel>

- Gefriertemperatur
  < sehr_tief, tief, nicht_tief>

- Wasserfilmhoehe, Windgeschwindigkeit
  <nicht_hoch, hoch, sehr_hoch>

- Belagsfeuchte
  <wenig, hoch, sehr_hoch>

- Taudiff
  <stark_neg, neg, null, pos, stark_pos >

- Gefrierdiff
  <neg_viel, negativ, null, positiv, pos_viel>

- LetAen (Letzte Änderung der Temperatur), MitAen (Mittlere Änderung, fünf Werte)Tempkorrektur
  <neg_sehr_viel, neg_viel, neg_wenig, null, pos_wenig, pos_viel, pos_sehr_viel>

- Feuchte
  <viel_weniger, weniger, gleich, hoeher, viel_hoeher>

Der Ausgangswert des Systems ist:

- Glaettewahrscheinlichkeit
  <unwahrscheinlich, moeglich, sicher>

Die endgültige Wissensbasis sieht wie folgt aus:

Wenn Taudiff stark_neg dann Feuchte viel_hoeher 1.0
Wenn Taudiff neg dann Feuchte hoeher 1.0
Wenn Taudiff null dann Feuchte gleich 1.0
Wenn Taudiff pos dann Feuchte weniger 1.0
Wenn Taudiff stark_neg dann Feuchte viel_weniger 1.0
Wenn Sonnenstand Sommer und Zeit Tag dann Feuchte weniger 0.0
Wenn Sonnenstand Sommer und Zeit Nacht dann Feuchte hoeher 0.0
Wenn Sonnenstand Winter dann Feuchte gleich 0.0
Wenn Fahrbahntemperatur sehr_warm dann Feuchte viel_hoeher 0.0
Wenn Fahrbahntemperatur warm dann Feuchte hoeher 0.0
Wenn Fahrbahntemperatur null dann Feuchte gleich 0.0
Wenn Fahrbahntemperatur kalt dann Feuchte weniger 0.0
Wenn Fahrbahntemperatur sehr_kalt dann Feuchte viel_weniger 0.0
Wenn Sonnenstand Sommer und Zeit Tag dann Tempkorrektur pos_sehr_viel 1.0
Wenn Sonnenstand Winter und Zeit Tag dann Tempkorrektur neg_sehr_viel 1.0
Wenn Sonnenstand Sommer und Zeit Nacht dann Tempkorrektur null 1.0
Wenn Sonnenstand Winter und Zeit Nacht dann Tempkorrektur neg_sehr_viel 1.0
Wenn LetAen neg_sehr_viel und MitAen neg_sehr_viel dann Tempkorrektur neg_sehr_viel 1.0
Wenn LetAen neg_sehr_viel und MitAen neg_viel dann Tempkorrektur neg_sehr_viel 1.0
Wenn LetAen neg_sehr_viel und MitAen neg_wenig dann Tempkorrektur neg_sehr_viel 1.0
Wenn LetAen neg_sehr_viel und MitAen null dann Tempkorrektur neg_sehr_viel 1.0
Wenn LetAen neg_sehr_viel und MitAen pos_wenig dann Tempkorrektur neg_viel 1.0
Wenn LetAen neg_sehr_viel und MitAen pos_viel dann Tempkorrektur neg_wenig 1.0
Wenn LetAen neg_sehr_viel und MitAen pos_sehr_viel dann Tempkorrektur neg_wenig 1.0
Wenn LetAen neg_viel und MitAen neg_sehr_viel dann Tempkorrektur neg_sehr_viel 1.0
Wenn LetAen neg_viel und MitAen neg_viel dann Tempkorrektur neg_sehr_viel 1.0
Wenn LetAen neg_viel und MitAen neg_wenig dann Tempkorrektur neg_viel 1.0
Wenn LetAen neg_viel und MitAen null dann Tempkorrektur neg_viel 1.0
Wenn LetAen neg_viel und MitAen pos_wenig dann Tempkorrektur neg_wenig 1.0
Wenn LetAen neg_viel und MitAen pos_viel dann Tempkorrektur pos_viel 1.0
Wenn LetAen neg_viel und MitAen pos_sehr_viel dann Tempkorrektur pos_wenig 1.0
Wenn LetAen neg_wenig und MitAen neg_sehr_viel dann Tempkorrektur neg_sehr_viel 1.0
Wenn LetAen neg_wenig und MitAen neg_viel dann Tempkorrektur neg_viel 1.0
Wenn LetAen neg_wenig und MitAen neg_wenig dann Tempkorrektur neg_sehr_viel 1.0
Wenn LetAen neg_wenig und MitAen null dann Tempkorrektur neg_viel 1.0
Wenn LetAen neg_wenig und MitAen pos_wenig dann Tempkorrektur neg_viel 1.0
Wenn LetAen neg_wenig und MitAen pos_viel dann Tempkorrektur pos_wenig 1.0
Wenn LetAen neg_wenig und MitAen pos_sehr_viel dann Tempkorrektur pos_viel 1.0
Wenn LetAen null und MitAen neg_sehr_viel dann Tempkorrektur neg_sehr_viel 1.0
Wenn LetAen null und MitAen neg_viel dann Tempkorrektur neg_viel 1.0
Wenn LetAen null und MitAen neg_wenig dann Tempkorrektur neg_wenig 1.0
Wenn LetAen null und MitAen null dann Tempkorrektur null 1.0
Wenn LetAen null und MitAen pos_wenig dann Tempkorrektur pos_wenig 1.0

Wenn LetAen null und MitAen pos_viel dann Tempkorrektur pos_viel 1.0
Wenn LetAen null und MitAen pos_sehr_viel dann Tempkorrektur pos_sehr_viel 1.0
Wenn LetAen pos_wenig und MitAen neg_sehr_viel dann Tempkorrektur neg_viel 1.0
Wenn LetAen pos_wenig und MitAen neg_viel dann Tempkorrektur neg_wenig 1.0
Wenn LetAen pos_wenig und MitAen neg_wenig dann Tempkorrektur pos_wenig 1.0
Wenn LetAen pos_wenig und MitAen null dann Tempkorrektur pos_wenig 1.0
Wenn LetAen pos_wenig und MitAen pos_wenig dann Tempkorrektur pos_sehr_viel 1.0
Wenn LetAen pos_wenig und MitAen pos_viel dann Tempkorrektur pos_viel 1.0
Wenn LetAen pos_wenig und MitAen pos_sehr_viel dann Tempkorrektur pos_sehr_viel 1.0
Wenn LetAen pos_viel und MitAen neg_sehr_viel dann Tempkorrektur neg_wenig 1.0
Wenn LetAen pos_viel und MitAen neg_viel dann Tempkorrektur pos_viel 1.0
Wenn LetAen pos_viel und MitAen neg_wenig dann Tempkorrektur pos_wenig 1.0
Wenn LetAen pos_viel und MitAen null dann Tempkorrektur pos_viel 1.0
Wenn LetAen pos_viel und MitAen pos_wenig dann Tempkorrektur pos_viel 1.0
Wenn LetAen pos_viel und MitAen pos_viel dann Tempkorrektur pos_sehr_viel 1.0
Wenn LetAen pos_viel und MitAen pos_sehr_viel dann Tempkorrektur pos_sehr_viel 1.0
Wenn LetAen pos_sehr_viel und MitAen neg_sehr_viel dann Tempkorrektur pos_wenig 1.0
Wenn LetAen pos_sehr_viel und MitAen neg_viel dann Tempkorrektur pos_wenig 1.0
Wenn LetAen pos_sehr_viel und MitAen neg_wenig dann Tempkorrektur pos_viel 1.0
Wenn LetAen pos_sehr_viel und MitAen null dann Tempkorrektur pos_sehr_viel 1.0
Wenn LetAen pos_sehr_viel und MitAen pos_wenig dann Tempkorrektur pos_sehr_viel 1.0
Wenn LetAen pos_sehr_viel und MitAen pos_viel dann Tempkorrektur pos_sehr_viel 1.0
Wenn LetAen pos_sehr_viel und MitAen pos_sehr_viel dann Tempkorrektur pos_sehr_viel 1.0
Wenn Lufttemperatur sehr_warm dann glatt unwahrscheinlich 0.5
Wenn Fahrbahntemperatur kalt und Gefrierdiff neg_viel dann glatt sicher 1.0
Wenn Fahrbahntemperatur sehr_kalt und Gefrierdiff neg_viel dann glatt sicher 1.0
Wenn Fahrbahntemperatur kalt und Gefrierdiff negativ dann glatt sicher 1.0
Wenn Fahrbahntemperatur sehr_kalt und Gefrierdiff negativ dann glatt sicher 1.0
Wenn Fahrbahntemperatur warm dann glatt unwahrscheinlich 0.3
Wenn Fahrbahntemperatur sehr_warm dann glatt unwahrscheinlich 0.5
Wenn Bodentiefentemperatur1 kalt dann glatt moeglich 0.3
Wenn Fahrbahnzustand feucht und Fahrbahntemperatur kalt dann glatt sicher 0.3
Wenn Fahrbahnzustand nass und Fahrbahntemperatur kalt dann glatt sicher 0.3
Wenn Fahrbahnzustand feucht und Fahrbahntemperatur sehr_kalt dann glatt sicher 0.3
Wenn Fahrbahnzustand nass und Fahrbahntemperatur sehr_kalt dann glatt sicher 0.3
Wenn Fahrbahnzustand Glatteis dann glatt sicher 1.0
Wenn Fahrbahnzustand Eisglaette dann glatt sicher 1.0
Wenn Fahrbahnzustand Reif dann glatt sicher 1.0
Wenn Fahrbahnzustand feucht und Lufttemperatur null dann glatt sicher 0.3
Wenn Fahrbahnzustand feucht und Lufttemperatur kalt dann glatt sicher 0.3
Wenn Fahrbahnzustand feucht und Lufttemperatur sehr_kalt dann glatt sicher 0.3
Wenn Fahrbahnzustand trocken dann glatt unwahrscheinlich 1.0
Wenn Belagsfeuchte sehr_hoch und Fahrbahntemperatur sehr_kalt dann glatt sicher 0.3
Wenn Belagsfeuchte hoch und Fahrbahntemperatur sehr_kalt dann glatt sicher 0.3
Wenn Belagsfeuchte hoch und Fahrbahntemperatur kalt dann glatt moeglich 0.5

Wenn Belagsfeuchte hoch und Fahrbahntemperatur null dann glatt unwahrscheinlich 0.3
Wenn Belagsfeuchte wenig und Fahrbahntemperatur sehr_kalt dann glatt unwahrscheinlich 0.3
Wenn Belagsfeuchte wenig und Fahrbahntemperatur kalt dann glatt unwahrscheinlich 0.3
Wenn Belagsfeuchte wenig und Fahrbahntemperatur null dann glatt unwahrscheinlich 0.3
Wenn Wasserfilmhoehe sehr_hoch und Fahrbahntemperatur sehr_kalt dann glatt sicher 0.3
Wenn Wasserfilmhoehe sehr_hoch und Fahrbahntemperatur kalt dann glatt sicher 0.3
Wenn Wasserfilmhoehe sehr_hoch und Fahrbahntemperatur null dann glatt moeglich 0.3
Wenn Wasserfilmhoehe hoch und Fahrbahntemperatur sehr_kalt dann glatt sicher 0.3
Wenn Wasserfilmhoehe hoch und Fahrbahntemperatur kalt dann glatt moeglich 0.3
Wenn Wasserfilmhoehe hoch und Fahrbahntemperatur null dann glatt moeglich 0.3
Wenn Restsalzgehalt viel und Fahrbahntemperatur sehr_kalt dann glatt moeglich 0.3
Wenn Restsalzgehalt viel und Fahrbahntemperatur kalt dann glatt unwahrscheinlich 0.3
Wenn Restsalzgehalt viel und Fahrbahntemperatur null dann glatt unwahrscheinlich 0.3
Wenn Restsalzgehalt mittel und Fahrbahntemperatur sehr_kalt dann glatt sicher 0.3
Wenn Restsalzgehalt mittel und Fahrbahntemperatur kalt dann glatt moeglich 0.3
Wenn Restsalzgehalt mittel und Fahrbahntemperatur null dann glatt moeglich 0.3
Wenn Restsalzgehalt wenig und Fahrbahntemperatur kalt dann glatt sicher 0.3
Wenn Restsalzgehalt wenig und Fahrbahntemperatur sehr_kalt dann glatt sicher 0.3
Wenn Restsalzgehalt wenig und Fahrbahntemperatur null dann glatt moeglich 0.3
Wenn Niederschlag vorhanden und Bodentiefentemperatur1 sehr_kalt dann glatt sicher 0.3
Wenn Niederschlag vorhanden und Bodentiefentemperatur1 kalt dann glatt moeglich 0.3
Wenn Niederschlag vorhanden und Fahrbahntemperatur sehr_kalt dann glatt sicher 0.3
Wenn Niederschlag vorhanden und Fahrbahntemperatur kalt dann glatt sicher 0.3
Wenn Niederschlag vorhanden und Fahrbahntemperatur null dann glatt moeglich 0.3
Wenn Luftfeuchte viel und Fahrbahntemperatur kalt dann glatt sicher 0.1
Wenn Luftfeuchte viel und Fahrbahntemperatur sehr_kalt dann glatt sicher 0.3
Wenn Luftfeuchte viel und Lufttemperatur sehr_kalt dann glatt sicher 0.3
Wenn Luftfeuchte viel und Lufttemperatur kalt dann glatt moeglich 0.3
Wenn Luftfeuchte mittel und Fahrbahntemperatur sehr_kalt dann glatt sicher 0.3
Wenn Luftfeuchte mittel und Fahrbahntemperatur kalt dann glatt sicher 0.3
Wenn Luftfeuchte mittel und Fahrbahntemperatur null dann glatt moeglich 0.3
Wenn Luftfeuchte mittel und Lufttemperatur sehr_kalt dann glatt moeglich 0.3
Wenn Luftfeuchte mittel und Lufttemperatur kalt dann glatt moeglich 0.3
Wenn Luftfeuchte wenig und Fahrbahntemperatur sehr_kalt dann glatt moeglich 0.3
Wenn Luftfeuchte wenig und Fahrbahntemperatur kalt dann glatt unwahrscheinlich 0.3
Wenn Luftfeuchte wenig und Lufttemperatur sehr_kalt dann glatt unwahrscheinlich 0.3
Wenn Luftfeuchte wenig und Lufttemperatur kalt dann glatt unwahrscheinlich 0.3
Wenn Windgeschwindigkeit sehr_hoch und Belagsfeuchte hoch und Fahrbahntemperatur null dann glatt sicher 0.3
Wenn Windgeschwindigkeit sehr_hoch und Belagsfeuchte sehr_hoch und Fahrbahntemperatur null dann glatt sicher 0.3
Wenn Windgeschwindigkeit sehr_hoch und Belagsfeuchte hoch und Fahrbahntemperatur kalt dann glatt sicher 0.3
Wenn Windgeschwindigkeit hoch und Belagsfeuchte hoch und Fahrbahntemperatur null dann glatt sicher 0.3

Wenn Windgeschwindigkeit hoch und Belagsfeuchte sehr_hoch und Fahrbahntemperatur null
dann glatt sicher 0.3
Wenn Windgeschwindigkeit hoch und Belagsfeuchte hoch und Fahrbahntemperatur kalt
dann glatt sicher 0.3
Wenn Gefriertemperatur nicht_tief und Fahrbahntemperatur sehr_kalt dann glatt sicher 1.0
Wenn Gefriertemperatur nicht_tief und Fahrbahntemperatur null dann glatt moeglich 0.3
Wenn Gefriertemperatur nicht_tief und Fahrbahntemperatur warm dann glatt unwahrscheinlich0.6
Wenn Gefriertemperatur tief und Fahrbahntemperatur sehr_kalt dann glatt sicher 0.3
Wenn Gefriertemperatur tief und Fahrbahntemperatur kalt dann glatt moeglich 0.3
Wenn Gefriertemperatur tief und Fahrbahntemperatur null dann glatt unwahrscheinlich 0.3
Wenn Gefriertemperatur tief und Fahrbahntemperatur warm dann glatt unwahrscheinlich 0.3
Wenn Gefriertemperatur sehr_tief und Fahrbahntemperatur sehr_kalt dann glatt moeglich 0.3
Wenn Gefriertemperatur sehr_tief und Fahrbahntemperatur kalt dann glatt unwahrscheinlich 0.3
Wenn Gefriertemperatur sehr_tief und Fahrbahntemperatur null dann glatt unwahrscheinlich 0.3
Wenn Gefriertemperatur sehr_tief und Fahrbahntemperatur warm dann glatt unwahrscheinlich 0.3
Wenn Taudiff stark_neg und Fahrbahntemperatur sehr_kalt dann glatt sicher 0.6
Wenn Taudiff neg und Fahrbahntemperatur sehr_kalt dann glatt sicher 0.6
Wenn Taudiff null und Fahrbahntemperatur sehr_kalt dann glatt moeglich 0.3
Wenn Taudiff pos und Fahrbahntemperatur sehr_kalt dann glatt unwahrscheinlich 0.3
Wenn Taudiff stark_pos und Fahrbahntemperatur sehr_kalt dann glatt unwahrscheinlich 0.3
Wenn Taudiff stark_neg und Fahrbahntemperatur kalt dann glatt sicher 0.6
Wenn Taudiff neg und Fahrbahntemperatur kalt dann glatt sicher 0.3
Wenn Taudiff null und Fahrbahntemperatur kalt dann glatt moeglich 0.3
Wenn Taudiff pos und Fahrbahntemperatur kalt dann glatt unwahrscheinlich 0.3
Wenn Taudiff stark_pos und Fahrbahntemperatur kalt dann glatt unwahrscheinlich 0.3
Wenn Gefrierdiff neg_viel dann glatt sicher 1.0
Wenn Gefrierdiff negativ dann glatt sicher 1.0
Wenn Gefrierdiff null dann glatt moeglich 1.0
Wenn Gefrierdiff positiv dann glatt unwahrscheinlich 1.0
Wenn Gefrierdiff pos_viel dann glatt unwahrscheinlich 1.0

In Bild 5.16 ist der Auswertebildschirm dargestellt. In Bild 5.17 sind die Ergebnisse der Vorhersagen dargestellt. Die Prognose_10 ist der Verlauf der prognostizierten Zwischengröße Fahrbahntemperatur_1 ohne Adaption, Verlauf Prognose_20 zeigt die Ergebnisse nach der Adaption. In Bild 5.18 ist die Ausgangsgröße Glättewahrscheinlichkeit dargestellt.
Die Verläufe in den Bildern 5.17 und 5.18 sind jeweils um 90 Minuten verschoben, so dass ein Vergleich zwischen Prognose und tatsächlichem Wert möglich ist. Die realen Messwerte sind gestrichelt dargestellt.

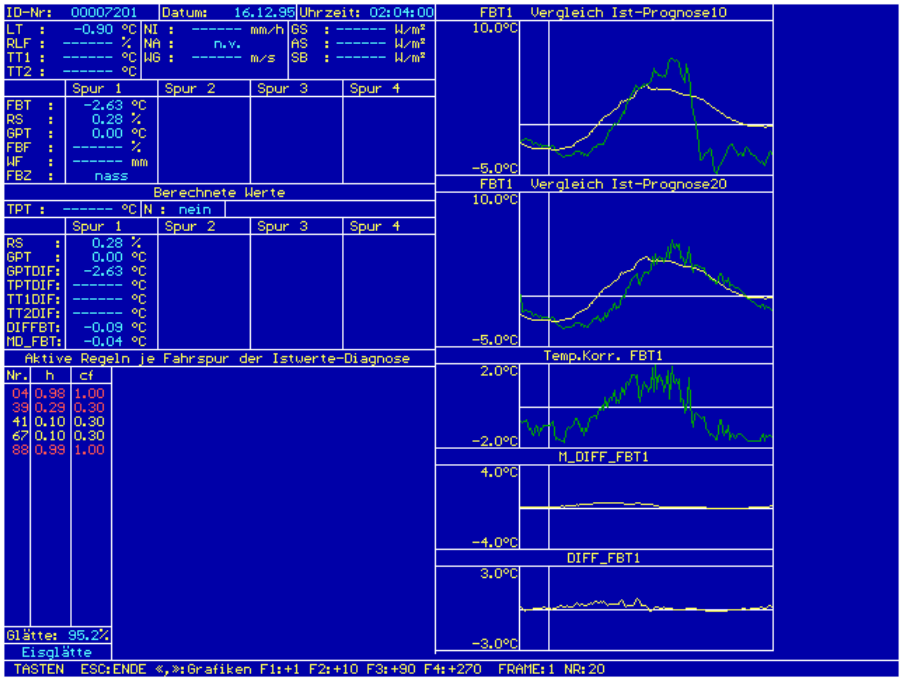


Bild 5.16: Ergebnisse vor (Ist_prognose10) und nach (Ist_prognose20) der Adaption

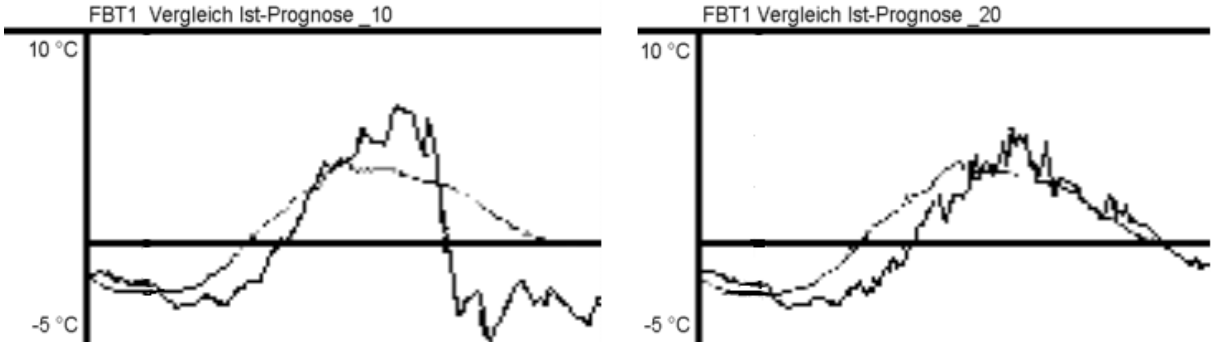


Bild 5.17: Ergebnisse der Vorhersagen mit und ohne Adaption

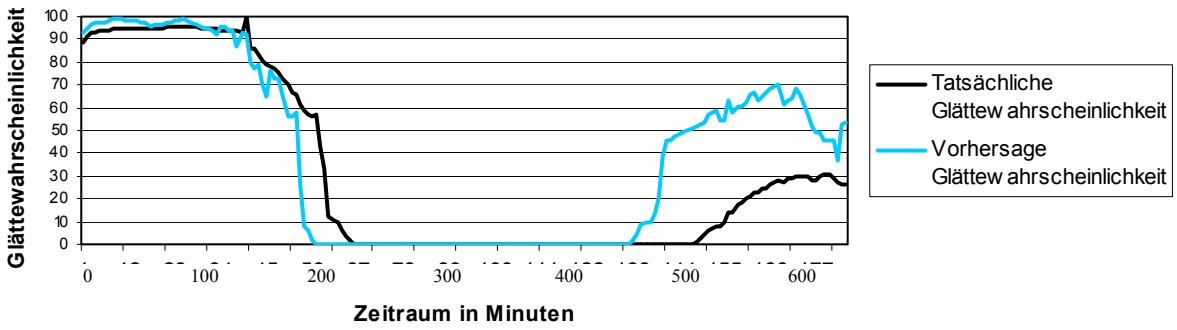


Bild 5.18: Ergebnisse der Vorhersagen für die Glättewahrscheinlichkeit

6 Zusammenfassung

In dieser Arbeit werden Systeme und Vorgehensweisen beschrieben, die es ermöglichen, das dynamische Verhalten eines Autofahrers zu erfassen und die Reaktionen zu optimieren. Das entwickelte System ist in der Lage auch Messwerte nachzubilden, die nicht direkt messbar, aber für eine effektive Weiterverarbeitung sehr nützlich und manchmal sogar notwendig sind.

Um dies zu erreichen, wurde ein Adaptionsverfahren entwickelt und getestet, das unter Verwendung von Gewichtungswerten die Wissensbasis optimiert. Des Weiteren ist im Rahmen dieser Arbeit ein Verfahren entwickelt worden, das in der Lage ist, die Wissensbasis eines Fuzzy-Systems mit Hilfe von genetischen Algorithmen zu erweitern und zu optimieren. Es ist hierdurch nicht unbedingt notwendig, sämtliche Zusammenhänge zwischen den vorhandenen Werten zu kennen und im Vorfeld zu beschreiben.

Es wurde auch der Lernalgorithmus für Neuronale Fuzzy-Netze mit Hilfe von α-Schnitten mit einem genetischen Algorithmus erweitert. Dies ermöglicht ein schnelleres Lernen des Netzes und die Ermittlung der optimalen Ergebnisse, da der Algorithmus in der Lage ist, nicht nur in ein lokales Minimum zu laufen, außerdem kann er auf Grund der Mutation auch unter Umständen das globale Minimum des Lernalgorithmus finden.

Es wurde gezeigt, dass sich mit Hilfe der beschriebenen Methoden und Vorgehensweisen Fehler in aufgenommen Messwerten sehr gut kompensieren lassen. Des Weiteren hat es sich gezeigt, dass ein intelligentes System auch in der Lage ist, nicht direkt messbare Werte zu ermitteln und mit diesen Werten hinreichend gute Prognosen zu erstellen. Dies wurde in dieser Arbeit bei der Glätteprognose anschaulich gezeigt. Die Implementierung des Systems zeigt, dass die Prognosegenauigkeit bei 95% liegt. Der Grund dafür, dass dieser Wert nicht noch höher liegt, ist beispielsweise, dass plötzlich einsetzender Regen nicht über einen bestimmten Zeitraum voraus gesagt werden kann. Das System liefert in diesem Zustand aber die Meldung *Eisgefahr bei zunehmender Feuchte*, so dass der Benutzer in der Lage ist, sich auf die Situation noch rechtzeitig einzustellen. Das System ist ebenfalls in der Lage, diese Genauigkeit auch nur mit wenigen Einstellungen selber zu erreichen. Durch die Adaption passt sich das System an die vorherrschenden Begebenheiten optimal an. Es ist auch in der Lage die Temperaturdynamik an den jeweiligen Einsatzorten und den Salzabrieb

selbständig und hinreichend genau zu ermitteln. Versuche mit Werten der Firma Micks aus dem Projekt SWIS BAWÜ haben gezeigt, dass das System bis 270 Minuten Prognosezeit noch bei einer Prognosegenauigkeit von 90% liegt.

Bei dem realisierten System für die Fahrzustandsbestimmung liegt die Genauigkeit bei 75%. Dieser Wert ist im Laufe der Arbeit durch Einbeziehung des Fahrerverhaltens auf 86% erhöht worden. Für eine Verwendung des Wertes reicht dieses aus. Eine Adaption als Ersatz für den Editor, die im Laufe der Arbeit realisiert wurde, brachte keine signifikante Verbesserung. Dies war auch zu erwarten, da die adaptierbaren Werte hauptsächlich durch das Benutzerprofil abgedeckt sind.

Die Probleme beginnen in dem Moment, wo mehrere Benutzer ein und dasselbe System verwenden. Wird für einen der Benutzer ein optimiertes System erstellt, kann dies für einen anderen Benutzer ein vollkommen unzureichendes Profil darstellen.

Die hier gewählten Ansätze für das Erkennen des Fahrerverhaltens sind in der Lage, die nächsten Reaktionen hinreichend genau vorauszusagen. Obwohl beide Verfahren sehr unterschiedlich sind, erzeugen sie jedoch annähernd gleiche Ergebnisse. Die resultierenden Zugehörigkeitsfunktionen sind bis auf kleine Unterschiede gleich.

Ein Test unter realen Bedingungen zeigt, dass zu 82% die Reaktion richtig vorhergesagt werden konnte. Dass hier keine höheren Werte möglich sind, liegt daran, dass bestimmte Verkehrssituationen nicht vorhersagbar sind. Hierdurch wird allerdings die Verwendbarkeit als zusätzliche Information zur Motorsteuerung nicht beeinträchtigt, da für solche Fälle sowieso übergeordnete Regeln greifen müssen. Als Beispiel ist hier ein plötzlicher Kickstart für einen Überholvorgang zu nennen.

Abschließend ist zu sagen, dass die ermittelten Verfahren nicht nur auf diese Anwendungen beschränkt sind. In [20, 22] wird eine Realisierung vorgeschlagen, die den Einsatz solcher Systeme zur Optimierung von Prozessbussystemen vorschlägt. Diese Realisierung steht zurzeit zwar noch in der Entwurfsphase, es sind jedoch schon viel versprechende Ergebnisse erzielt worden.

7 Literatur

[1] Bonfig, K.W., u. a. „Fuzzy-Logik in der industriellen Automatisierung“, Expert Verlag, Deutschland, (1992)

[2] Aliev R., Bonfig, K.W., Aliew, F. „Messen, Steuern, Regeln mit Fuzzy-Logik“, Franzis Verlag, Deutschland, (1994)

[3] Kosko, B. „Neural Networks and Fuzzy Systems“, Prentice Hall, USA, (1992)

[4] Yager R.R., u.a. „ Fuzzy Sets, Neural Networks and Soft Computing, Von Nostrad Reinhold, New York, USA, (1994)

[5] Wieland F., u.a : „A Fuzzy Expert System for railway rolling excavator“; Proceedings Icafs 94 , Tabriz, Iran, (1994)

[6] Wieland F., : „ Boundary filling level control with inacurate sensors in areas with danger of explosion by the using of Fuzzy Control “; Proceedings Icafs 94, Tabriz, Iran, (1994)

[7] Wieland F., u.a : „Description of a Fuzzy Expert System for very large rail bound mining devices in surface work mining area“; Proceedings Ifac 95, München, Deutschland, (1995)

[8] Wieland F., u.a. : „Contact free level control with inacurate sensors by using Fuzzy Control techniques on a µ-Controller“; Proceedings Eufit 95, Aachen, Deutschland, (1995)

[9] Wieland F., u.a : „Description of a Fuzzy Expert System for very large rail bound mining devices in surface work mining area“; Proceedings Icafs 96 , Siegen, Deutschland, (1996)

[10] Wieland F.,: „ Adaptive Control System with Neuro-Fuzzy-Genetic Approach“; Proceedings Icafs 96, Siegen, Deutschland, (1996)

[11] Wieland F., u.a (Hrsg): ICAFS 96 Proceedings, Siegen, Deutschland, (1996)

[12] Wieland F., u.a : „Optimierung ungenauer Sensoren mit Hilfe der Methoden des Soft-Computings“, Meßtechnik und Messignalverabeitung, Expert Verlag, Deutschland, (1996)

[13] Wieland F., u.a : „Description of a Fuzzy Expert System for very large rail bound mining devices in surface work mining area“; Proceedings Icafs 98 , Wiesbaden, Deutschland, (1998)

[14] Wieland F.,: „ Adaptive Control System with Neuro-Fuzzy-Genetic Approach“; Proceedings Icafs 98, Wiesbaden, Deutschland, (1998)

[15] Wieland F., u.a : „Ermittlung und Verarbeitung nicht direkt messbarer Signale mit Methoden des Soft-Computings“, Sensoren und Feldbussystem, b-Quadrat Verlags GmbH, Deutschland, (1998)

[16] Wieland F., u.a : „Ansätze zur Messwertverarbeitung und Messwertermittlung mit Hilfe des Soft-Computings“, Meßtechnik und intelligente Messtechnikkonzepte, b-Quadrat Verlags GmbH, Deutschland, (1998)

[17] Wieland F., u.a : „A Soft Computing Control Approach for Future Processbus-Systems“; Proceedings Eufit 99, Aachen, Deutschland, (1999)

[18] Bonfig, K.W. „Neuro-Fuzzy“, Expert Verlag, Deutschland, (1995)

[19] Bonfig, K.W. „Soft-Computing “, Verlag Technik, Deutschland, (2000)

[20] Wieland, F. u.a.: „A Soft Computing Supervising System for Processbus System“ Proceedings Icafs 2000, Siegen, Deutschland, (2000)

[21] Wieland, F. u.a.: „A Soft Computing Measurement and Decision System for Engine-Control“ Proceedings Icafs 2000, Siegen, Deutschland, (2000)

[22] Wieland, F. u.a.: „A New Method for Supervising of Processbus System“ Proceedings WAC 2000, Maui, Hawaii, USA, (2000)

[23] Wieland, F. u.a.: „Soft Computing in Measurement Systems“ Proceedings WAC 2000 , Maui, Hawaii, USA, (2000)

[24] Auweiler, W.: „Entwurf, Entwicklung und Programmierung eines Neuro-Fuzzy-Controllers mit einem genetischen Lernalgorithmus in C++ für die Erkennung von Benutzerprofilen“ Diplomarbeit am Institut für Meßtechnik, Deutschland, (1999)

[25] Davidor,Y.: „Gentic algorithms and robotics“, World Scientific Publishing, Singapore, (1990)

[26] Isermann, R.: „Digital Control Systems“, Springer Verlag, Heidelberg, Deutschland, (1987)

[27] Hart, M.: „Auswertung direkter Brennrauminformationen am Verbrennungsmotor mit estimationstheoretischen Methoden, Dissertation Universität Siegen, Deutschland, (1999)

[28] Schedler. K. : „IceForecast“, interne Beschreibung der Funktionsweise des Prognosemoduls der Firma Micks, Obersdorf, Deutschland, (1995)

[29] Schedler. K. : „Forecast für Stausituationen“, interne Beschreibung der Funktionsweise des Prognosemoduls der Firma Micks, Obersdorf, Deutschland, (1996)

[30] Aliev R.A.: „Fuzzy Expert systems in Soft Computing“, Uni Press, New Jersey, USA, (1994)

[31] Mizumoto M.: „Fuzzy Conditional Inference under Max-Composition“ Information Sciences, S.183-209, USA, (1982)

[32] Bandler W., Kohout L.: „Fuzzy power sets and fuzzy implications operators“ Fuzzy sets and systems, S.13-30, USA, (1980)

[33] Rescher A. : „Many-Valued Logic“, McGraw-Hill, New York, USA, (1969)

[34] Aliev R.A.,Zerkovny A.E.,Mamedova G.A.:Upravlenie proisvodstvom pri netschetkoy ischodnoy informaziji (Produktionssteuerung bei unscharfen Eingangswerten), S.240 Moskau, (in Russ.), Russland, (1991)

[35] Mamdani E.H., „Applications of fuzzy logic to approximate reasoning using linguistic systems“, IEEE Transactions on Computer, S.1182-1191, USA, (1977)

[36] Baldwin J.F.,Pilsworth B.W. „A model of fuzzy reasoning through multivalued logic and set theory“, International Journal on Man-Machine Studies, S.351-380, USA, (1979)

[37] Michalewicz Z.: Genetic Algorithms+Data Structures = Evolution Programs, Springer Verlag, Berlin, Deutschland, (1996)

[38] Aliev R.A, Abilov Y.A, Nasirov I.M.: „Theory of possibility and group pripciple in genetic algorithms in Proc. Int. Conf. on Application of Fuzzy Systems, Tabriz, Iran, (1994)

[39] Ackley D.H.: „A connectionist algorithm for genetic search“, Proceeding of Int. Conf on Genetic Algorithms and Their Aplication, S.121-135, USA, (1985)

[40] Goldberg D.E: Genetic Algorithms in Search, Optimization and Machine Learning. Reading, USA, (1989)

[41] Aarts E.H., Korst J.: Simulated annealing and Boltzmann machines, Chichester, UK, (1989)

[42] Laarhoven P.J., Van M., Aarts E.H.: Simulated annealing: theory and applications, Holland, (1987)

[43] Niedringhaus W.P.: „Maneuver Option Manager: Automated Simplification of Complex Air Traffic Problems“, IEEE Transactions on systems, man, and cybernetics. Vol.22, USA, (1991)

[44] Walley P.: Measures of uncertainty in expert systems, Artifical intelligence 83, S. 1-58, USA, (1996)

[45] Wilson N., Moral S.: A logical view of probability: A., Proc. Eleventh European Conf. On Artificial Intelligence, London, S. 386-390, UK, (1994)

[46] Hartmann, M.: „Fuzzy-Logik gestützte Stauprognose auf der Basis fahrzeugeigener Daten“, Diplomarbeit am Institut für Meßtechnik, Siegen, Deutschland, (1999)

[47] Seifert, B: „ i3000 Presentation“, Cosworth Technology, USA, (2001)

[48] Hopfield, J.J.: „Neural Network and Physical Systems with Emergent Collective Computational Abilities“, Proc. of the National Academy of Science 79, S.2554-2559, USA, (1982)

[49] Kohnen, T. : „Content-Adressable Memories“, Berlin-Heidelberg-Wien: Springer-Verlag, 2nd ed., Deutschland, (1987)

[50] Grossberg, S.: „Neural Networks and Natural Intelligence“ Cambridge: MIT Press, USA, (1988)

[51] Gellert, W., u.a.: „Kleine Enzyklopädie Mathematik“ Leipzig, Bibliographisches Institut, Deutschland, (1985)

[52] Rigoll, G. u.a.: „ Neuronale Netze“, Ostfildern, TAE-Unterlagen, Deutschland, (1992)

[53] Scherer, A.: „Neuronale Netze“, Braunschweig, Wiesbaden, Vieweg, Deutschland, (1997)

[54] Börcsök, J.: „Fuzzy Control“, Berlin, Verlag Technik, Deutschland, (2000)

[55] Zadeh, L.A.: „Fuzzy Sets“, Information and Control, 8, S. 338-353, Rolling Meadows, USA, (1965)