

Informatik
Hauptcampus

H O C H
S C H U L E
T R I E R

Titel der Arbeit

Bearbeiter: Bachvarov, Vladislav

Gruppe: GruppenID

Projektstudium

Ort, Abgabedatum

Kurzfassung

In der Kurzfassung soll in kurzer und prägnanter Weise der wesentliche Inhalt der Arbeit beschrieben werden. Dazu zählen vor allem eine kurze Aufgabenbeschreibung, der Lösungsansatz sowie die wesentlichen Ergebnisse der Arbeit. Ein häufiger Fehler für die Kurzfassung ist, dass lediglich die Aufgabenbeschreibung (d.h. das Problem) in Kurzform vorgelegt wird. Die Kurzfassung soll aber die gesamte Arbeit widerspiegeln. Deshalb sind vor allem die erzielten Ergebnisse darzustellen. Die Kurzfassung soll etwa eine halbe bis ganze DIN-A4-Seite umfassen.

Hinweis: Schreiben Sie die Kurzfassung am Ende der Arbeit, denn eventuell ist Ihnen beim Schreiben erst vollends klar geworden, was das Wesentliche der Arbeit ist bzw. welche Schwerpunkte Sie bei der Arbeit gesetzt haben. Andernfalls laufen Sie Gefahr, dass die Kurzfassung nicht zum Rest der Arbeit passt.

The same in english.

Inhaltsverzeichnis

1	Einleitung und Problemstellung	1
2	Unscharfe Daten und Fuzzy-Sets	2
2.1	Fuzzy-Sets	2
2.1.1	Repräsentation von Fuzzy-sets	2
2.2	Operationen auf Fuzzy-Sets	3
2.2.1	Durchschnitt	3
2.2.2	Vereinigung	4
2.2.3	Komplement	4
2.2.4	Zusammenfassung	5
2.3	Fuzzy-Systeme	5
2.3.1	Mamdani-Regler	5
2.3.2	Takagi-Sugeno-Kang-Modell	7
2.3.3	Fuzzifizierung und Defuzzifizierung	7
3	Neuro-Fuzzy-Systeme	9
3.1	Neuro-Fuzzy-Regler	9
3.1.1	Modelle für feste Lernaufgaben	9
3.1.2	Modelle mit verstärkendem Lernen	11
	Literaturverzeichnis	15
	Glossar	16

Einleitung und Problemstellung

Das Thema dieser Ausarbeitung ist Unschärfe Informationen, oder Fuzzy-Sets. Mit Fuzzy-Sets lassen sich schwammige Daten, wie "große Zahlen oder "mittlere" Temperatur, für Maschinen, insbesondere Computern, beschreiben. Diese Mengen können dann von so genannten Fuzzy-Systeme interpretiert werden. Damit zieht man entsprechend bestimmte logische Rückschlüsse bezüglich einer Eingabe. Als Beispiel könnte man die Farben von Tomaten. Der Mensch kann mit höher Richtigkeit entscheiden, welche Tomate denn reif ist. Für eine Maschine jedoch ist die Interpretation roher Information (Tomate ist Rot, also reif) nicht möglich. Mit den Fuzzy-Sets und Regeln kann man in dem System, dieses definieren.

Ziel dieser ist dann, unter Anwendung von Neuronalen Netzen, Parameter beliebiger Fuzzy-Modelle zu optimieren. Das würde heißen, dass bei der Mathematische Funktion $y = a * x_1 + b * x_2$ die Parametern a und b von dem neuronalen Netz automatisch angepasst werden, sodass sich der Erwartungswert y' bei den Eingaben x_1 und x_2 ergibt.

In dieser Ausarbeitung werden einige Ansätze vorgestellt. Schließlich wird eine Entscheidung getroffen, welcher Ansatz verwendet wird, Neuro-Fuzzy-Systeme aufzubauen.

Unscharfe Daten und Fuzzy-Sets

Einleitung hier...

2.1 Fuzzy-Sets

In der Literatur bezeichnet man das Verfahren zur Darstellung von unscharfen Daten als Fuzzy-Logic. Fuzzy-Logic unterscheidet sich von der klassischen Mengenlehre darin, dass Elemente graduell einer Menge gehören und nicht nur bivalente Zugehörigkeit erweisen können.

Zum Verdeutlichen betrachten wir die Menge M der reellen Zahlen, die viel größer sind als 1 [RCC⁺15].

$$M = \{x \mid x \in \mathfrak{R}, x \gg 1\} \quad (2.1)$$

Wird diese Menge mit der klassischen Logik modelliert, ergibt sich die Problematik: Der Vergleich "viel größer ist" mathematisch nicht eindeutig definiert. Auf die Grafik unten kann man die Modellierung mit klassischen Logik sehen.

Aus der Graphik kann man feststellen, dass alle Werte kleiner als 10 eine Zugehörigkeit von 0 und solche größer als 10 - eine Zugehörigkeit von 1 besitzen. Diese Repräsentation entspricht jedoch die Realität nicht so ganz. Es ist eindeutig, dass ja 9.9 als Wert schon größer als 1 ist, aber nach der Graphik wird das nicht klar. Würde man diese Menge in der Fuzzy-Logik darstellen, ergibt sich folgender Graph.

Hier beschreibt die Darstellung besser, wie die Zahlen im Vergleich zu 1 stehen. Die Zugehörigkeit nimmt Werte zwischen 0 und 1. Zum Beispiel der Wert 10 hat die Zugehörigkeit 0.15 und für Werte größer als 20 liefert die Funktion $\mu(x)$ einen Wert von 1 (volle Zugehörigkeit).

2.1.1 Repräsentation von Fuzzy-sets

[Fuzzy-Logik und Fuzzy-Control, Jörg Kahlert; Hubert Frank,]

[Fuzzy-Logik und Fuzzy-Control, Jörg Kahlert; Hubert Frank, Computational Intelligence]

2.2 Operationen auf Fuzzy-Sets

In dem Vorherigen Kapitel wurde auf die Wichtigkeit von Fuzzy-Logik, genau so wie Repräsentation von Fuzzy-Mengen eingegangen. Um nun unscharfe Informationen verarbeiten zu können, wie Schlüsse daraus zu ziehen oder mehrere Fuzzy-Mengen zu kombinieren, brauchen wir eine Reihe von Operatoren. Da es um Mengen geht, eignen sich die Durchschnitt-, Vereinigung- und Komplementbildung aus der klassischen Logik gut. Im folgenden Kapitel werden die einzelnen Operationen beschrieben [Computational Intelligence].

2.2.1 Durchschnitt

In der klassischen Logik ist der Durchschnitt durch einen Logischen-UND eingesetzt. In der klassischen Mengenlehre ist die Menge aller Elementen, die zu einer Menge M_1 und einer Menge M_2 gehören, als Schnittmenge definiert. Gegeben sei die Mengen:

$$M_1 = \{x \mid x \in \mathfrak{R}, 1 \leq x \leq 3\} \quad (2.2)$$

$$M_2 = \{x \mid x \in \mathfrak{R}, 2 \leq x \leq 4\} \quad (2.3)$$

Der Durchschnitt der beiden Mengen ergibt sich:

$$M_1 \cap M_2 = \{x \mid x \in \mathfrak{R}, 2 \leq x \leq 3\} \quad (2.4)$$

Graphisch dargestellt, ist die Schnittmenge der Bereich der in beiden Mengen enthalten ist.

Dies lässt sich direkt auf die Fuzzy-Mengen übertragen. Die UND-Verknüpfung für Fuzzy-Mengen ist als den Durchschnitt der Flächen unter den Graphen ihrer Zugehörigkeitsfunktion definiert. Aus mathematischer Sicht ist das, dass Minimum-Operator (MIN)

Definition 2.1. Seien μ_1 und μ_2 zwei Fuzzy-Mengen auf der Grundmenge G . Dann heißt:

$$\mu_1 \cap \mu_2 : G \rightarrow [0,1] \text{ mit } (\mu_1 \cap \mu_2)(x) = \min(\mu_1(x), \mu_2(x))$$

der **Durchschnitt** der Fuzzy-Mengen μ_1 und μ_2 .

Zur Veranschaulichen wird unten die Grafik angegeben. Da sind zwei Fuzzy-Sets dargestellt. Die zwei Ausdrücke, die betrachtet werden, sind *mittlere* und *hohe* Temperatur. Die Ergebnismenge repräsentiert alle *mittlere* und *hohe* Temperaturen.

2.2.2 Vereinigung

Nehmen wir ganz einfach die Definition der Vereinigung aus der klassischen Mengenlehre:

$$x \in M_1 \cup M_2 \Leftrightarrow x \in M_1 \vee x \in M_2$$

Ziemlich eindeutig und klar. Die Vereinigungsmenge enthält alle diese Elemente aus dem Grundbereich, die entweder in der Menge M_1 oder M_2 enthalten sind. Im nächsten Schritt wird dieser Operator an die Fuzzy-Mengen angepasst.

In der Mathematik ist dieser Operator als ODER-Verknüpfung angegeben. Die Anwendung der ODER-Operator auf Fuzzy-Mengen wird wie folgt dann definiert:

Definition 2.2. Seien μ_1 und μ_2 zwei Fuzzy-Mengen auf der Grundmenge G . Dann heißt:

$$\mu_1 \cup \mu_2 : G \rightarrow [0,1] \text{ mit } (\mu_1 \cup \mu_2)(x) = \text{MAX}(\mu_1(x), \mu_2(x))$$

die **Vereinigung** der Fuzzy-Mengen μ_1 und μ_2 .

Betrachten wir den selben Beispiel aus vorherigen Unterkapitel. Seien also wieder die Fuzzy-Mengen für "mittlere" und "hohe" Temperatur. Wenn wir den ODER-Operator auf die beiden Mengen anwenden, ergibt sich eine Ergebnismenge, die sich auf beiden Mengenflächen aufstellt. Dies wurde graphisch zunächst aufgezeichnet.

Betrachten wir nochmal die beiden Abbildungen (Vereinigung und Durchschnitt). Es erhebt sich die Frage, warum sind die nämlich so definiert und nicht wie folgt?:

Man erwartet ja, dass Elemente, die in beiden Mengen liegen (also das Element weist beide Eigenschaften hohe und mittlere Temperatur), auch eine Zugehörigkeit von 1 annehmen dürfen. Für die Praxis spielt das keine große Rolle [RCC⁺15].

Für die üblichen Operatoren gelten entsprechend die Eigenschaften auf Mengenoperationen, wie z.B. Assoziativität, Distributivität, Kommutativität usw. Ich gehe in dieser Ausarbeitung nicht weiter auf die einzelnen Eigenschaften.

Neben den Grundverknüpfungen gibt es eine weitere Sammlung von Verknüpfung-Operatoren. Diese sind z.B. :

1. Algebraisches Produkt: $(\mu_1 \mu_2) = \mu_1(x) \cdot \mu_2(x)$
2. direkte Summe: $(\mu_1 \oplus \mu_2) = \mu_1(x) + \mu_2(x) - \mu_1(x) \cdot \mu_2(x)$
3. abgeschnittene Differenz: $(\mu_1 \dot{-} \mu_2) = \text{MAX}(0, \mu_1(x) - \mu_2(x))$
4. abgeschnittene Summe: $(\mu_1 \hat{+} \mu_2) = \text{MIN}(1, \mu_1(x) + \mu_2(x))$
5. $(\mu_1 \div \mu_2) = \text{MIN}(\mu_1(x), 1 - \mu_2(x))$
6. ...

2.2.3 Komplement

Der dritte wichtige Operator ist das Komplement einer Menge. In der klassischen Mengenlehre ist dieser Operation ziemlich einfach anzuwenden. der Operator beschreibt die Negation einer Aussage, wenn wir jetzt über Wahrscheinlichkeit reden würden, die Gegenwahrscheinlichkeit, z.B. Die Wahrscheinlichkeit eine 6 zu würfeln

wäre $\frac{1}{6}$, die Gegenwahrscheinlichkeit, oder die Wahrscheinlichkeit etwas anderes als 6 zu würfeln wäre: $(1 - \frac{1}{6}) = \frac{5}{6}$.

Genau so übertragen wir diesen Operator auf die Fuzzy-Mengen. Das Komplement ist dann wie folgt definiert:

Definition 2.3. Sei μ eine Fuzzy-Menge auf der Grundmenge G . Dann heißt:

$$\mu^c : G \rightarrow [0,1] \text{ mit } (\mu^c)(x) = 1 - \mu(x)$$

die **Vereinigung** der Fuzzy-Mengen μ_1 und μ_2 .

Zur Veranschaulichung nehme ich wieder eine Fuzzy-Menge und wende die Operation auf diese. Ich wähle hier die Menge *hohe* Temperatur. Wir bekommen dann die Fuzzy-Menge *NICHT hohe* Temperatur. Die Graphik ist unten gezeigt:

2.2.4 Zusammenfassung

In diesem Kapitel habe ich die drei wichtigsten Operatoren auf Fuzzy-Mengen vorgestellt. Dies ist eine Vorbereitung auf die folgenden Kapiteln. Wir analysieren mehrere Fuzzy-Mengen und ziehen daraus bestimmte Schlüsse. Ein Beispiel dafür wäre: Wenn eine Tomate rot ist, dann ist die reif. Folgende Logische Aussagen können sehr einfach mit Fuzzy-Logic modelliert werden. In der Literatur taucht die Name Fuzzy-Regeln. Diese können dann zusammengestellt werden, um Fuzzy-Regel-Systeme aufzubauen. Die nächste Kapitel stellt Fuzzy-Systeme vor.

2.3 Fuzzy-Systeme

Ein Fuzzy-System, auch Fuzzy-Regler oder Fuzzy-Inferenz-System, ist ein bekannter Framework basierend auf Fuzzy-Mengen-Theorie, Fuzzy Wenn-Dann-Regeln und Fuzzy Reasoning. Konzeptionell besteht ein Fuzzy-System aus drei Grundteile - die Regelbasis, welche die Wenn-Dann-Regeln beinhaltet; die Datenbank(oder das Wörterbuch), die alle Zugehörigkeitsfunktionen definiert; und der Entscheidungsmechanismus, wer die Inferenz durchführt und einen angemessenen Schlussfolgerung erkundet.

Folglich lässt sich zwei Arten von Fuzzy-Systeme erläutern.

2.3.1 Mamdani-Regler

Der Mamdani-Regler wurde im Jahr 1975 von Mamdani auf der Basis einer Veröffentlichung von Zadeh aus der Anfang der siebziger Jahren entwickelt.

Der Mamdani-Regler ist eine endliche Menge von Wenn-Dann-Regeln R der Form:

$$R : \text{If } x_1 \text{ is } A_1 \text{ and ... and } x_n \text{ is } A_n \text{ then } y' \text{ is } B \quad (2.5)$$

In der Regel sind x_1, \dots, x_n die Eingangsgrößen und y' die Ausgabe. A_i und B sind linguistischen Werte. Jede Regel besteht aus zwei Teilen - Prämisse und

Konklusion. Jede Vorbedingung untersucht spezifische Eigenschaften, die der Eingangswert erfüllen muss. Anhand diese Merkmale schließt sich eine Konklusion.

Sei W die Menge aller linguistischen Konklusionen B_i , so dass $B_1, \dots, B_m \in W$ gilt. Der Regler kann als eine n -Stellige Funktion mit $f : \mathbb{G}^n \mapsto W$ dargestellt werden. Die Funktion ordnet eine Eingabe zu einem linguistischen Term B_i .

$$f(x_1, \dots, x_n) \approx \begin{cases} B_1 & \text{falls } x_1 \text{ is } A_1^{(1)} \text{ und } \dots, \text{ und } x_n \text{ is } A_n^{(1)} \\ \vdots \\ B_m & \text{falls } x_1 \text{ is } A_1^{(m)} \text{ und } \dots \text{ und } x_n \text{ is } A_n^{(m)} \end{cases} \quad (2.6)$$

In der Funktion gibt es genau so viele Ausgaben, wie es Regeln gibt - m .

Mit einem Mamdani-Regler können viele Probleme aus der reellen Welt definiert werden. Als kleines Beispiel eignet sich die Farben der Tomaten. Wir betrachten die Farbe als Eingabe. Die Ausgabe würde uns sagen, wie reif eine Tomate ist. Wir teilen unsere Eingangsgrößen für die Farbe in drei Mengen: rot, gelb, grün. Daraus lassen sich entsprechend 3 Fuzzy-Mengen für die Eingabewerte definieren. Als Ausgabe können wir wieder für die Reife einer Tomate drei Mengen definieren: reif, halbreif oder unreif. Das sind natürlich unsere Fuzzy-Mengen für die Konklusion. Auf dieser Weise haben wir einen Mamdani-Fuzzy-Model mit 3 Fuzzy-Sets für Farbe und aus 3 Regeln besteht. Die Regelbasis sieht wie folgt aus:

1: if x is rot, then y is reif.

R_2 : if x is gelb, then y is halbreif.

R_3 : if x is grün, then y is unreif.

Eine Tabelle für die Fuzzy-Relationen ist unten gegeben

$x \setminus y$	unreif	halbreif	reif
grün	1	0	0
gelb	0	1	0
rot	0	0	1

Tabelle 2.1. Beispiel für Tomaten

Wie man sieht die Tabelle ist ziemlich einfach zu lesen, wenn die Tomate größtenteils grün ist, wird immer angenommen dass sie unreif ist und nichts weiteres. Man könnte entsprechend die Fuzzy-Mengen aus der Konklusion verfeinern. Das würde bedeuten, dass die Mengen überlappen. Unsere Tabelle könnte dann wie folgt aussehen:

Abhängig davon wie man die Ausgangsmengen definiert, wie weit die Mengen überlappen, ergibt sich eine unterschiedliche Interpretation der Werte. Je mehr Fuzzy-Sets für eine Größe(Maß) definiert sind, desto besser könnte ihr Zustand repräsentiert werden. Mit der Anzahl der Fuzzy-Sets steigt die Komplexität eines Systems proportional.

Den oberen kleinen Beispiel zeigt wie einfach Mamdani-Modelle zu modellieren sind. Mamdani-Reglern finden in der Praxis öfters Einsetzung.

$x \setminus y$	unreif	halbreif	reif
grün	1	0.5	0
gelb	0.3	1	0.3
rot	0	0.5	1

Tabelle 2.2. Beispiel für Tomaten

2.3.2 Takagi-Sugeno-Kang-Modell

Das Takagi-Sugeno-Kang-Modell ist dem Mamdani-Modell sehr ähnlich. TSK-Regler verwenden Regeln der Form:

$$R: \text{ If } x_1 \text{ is } \mu_R^{(1)} \text{ and } \dots \text{ and } x_n \text{ is } \mu_R^{(n)} \text{ then } y = f_R(x_1, \dots, x_n). \quad (2.7)$$

In der Prämisse ist kein Unterschied zu erkennen. Die Besonderheit des TSK-Modells ergibt sich in der Konklusion. Da steht eine Funktion, anstatt eine Fuzzy-Menge. In der Konklusion kann im allgemeinen Fall jede beliebige Funktion f in der Eingabe von x berechnen. Normalerweise wird f eingeschränkt, eine linear Kombination der Eingangsgrößen x zu sein:

$$f(x_1, \dots, x_n) = p_0 + p_1 \cdot x_1 + \dots + p_n \cdot x_n, \quad (2.8)$$

wo p_0, \dots, p_n die Parametern der Konklusionsfunktion sind. Für geeignet ausgewählten Parameterwerten modelliert das TSK-Modell eine Mathematische Funktion, dies kann z.B. die Quadratische sein. Für den Fall, dass die Parametern p_1, \dots, p_n den Wert 0 haben, erhält man einen Mamdani-Modell mit scharfer Ausgabe. Solche Modelle heißen in der Literatur auch **zero-order Sugeno-Fuzzy-Modelle** [AM01, Han98, JCE97]. Als Beispiel kann folgende Regelbasis betrachtet werden:

$$R_1: \text{ If } x \text{ is 'sehr klein' then } y = 0 \quad (2.9)$$

$$R_2: \text{ If } x \text{ is 'klein' then } y = 1 \quad (2.10)$$

$$R_3: \text{ If } x \text{ is 'groß' then } y = 2 \quad (2.11)$$

$$R_4: \text{ If } x \text{ is 'sehr groß' then } y = 3. \quad (2.12)$$

Die Fuzzy-Sets sind 'sehr klein', 'klein', 'groß', 'sehr groß'. Entsprechend werden die Konklusionen daran zugeordnet.

2.3.3 Fuzzifizierung und Defuzzifizierung

Der Vorgang eines Modells zerfällt in zwei Teilen - Fuzzifizierung und Defuzzifizierung. Die Fuzzifizierung findet zuerst statt.

Die Ausgabe eines Fuzzy-Modells wird auch als Defuzzifizierung. Davor findet die Fuzzifizierung statt.

Bei Regelaktivierung unterscheidet man zwei Fällen. Die Eingabe ergibt sich zu einer völligen Aktivierung. Bei dem Fall liefert das Modell ganz normal den Wert der entsprechenden Konklusionsfunktion. Bei nicht volle Aktivierung mehreren Regeln ergibt sich die Ausgabe über folgende Formel:

Neuro-Fuzzy-Systeme

3.1 Neuro-Fuzzy-Regler

3.1.1 Modelle für feste Lernaufgaben

(Muss Änderungen unterliegen)

Neuro-Fuzzy-Modelle für feste Lernaufgaben versuchen, Fuzzy-Mengen und, bei TSK-Modellen, die Parameter der Ausgabefunktion unter Einreichung einer Mengen von Ein-/Ausgabe-Tupeln zu optimieren. Diese Modelle sind genau dann sinnvoll, wenn schon eine Fuzzy-Regelbasis vorliegt. Die Regelbasis unterliegt infolge des Lernens eine Verarbeitung, die als Ziel eine Optimierung hat.

Ein weiteres Anwendungsbeispiel ist bei bereits existierender Regelbasis, die mit einer Neuen ausgetauscht werden. Falls die Basis schon errechnete Werte geliefert hat, können diese Zusammen mit den zugehörigen Eingabewerten dem Neuro-Fuzzy-System gegeben werden. Am Ende erhält man eine optimierte Fuzzy-Regel-Basis, die die alte Basis ersetzt“.

Falls es keine angemessene Lernaufgabe bereits gegeben ist, dann eignet sich dieses Verfahren nicht. Es existieren natürlich Ansätze, die es ermöglichen, einen initialen Regelbasis aus Eingabedaten zu erstellen. Ein solcher Verfahren wird später noch vorgestellt.

Folglich wird einen Beispiel von einem Modell, das sich für feste Lernaufgaben eignet, vorgestellt. Es geht nämlich um das ANFIS-Modell.

Das ANFIS-Modell

Im Frühjahr von 1993 wurde das Neuro-Fuzzy-System ANFIS (Adaptive Neuro-Fuzzy Inference System) entwickelt. Heutzutage ist dieser Ansatz viel verbreitet, und das Modell wurde in mehrere Software-Pakete schon eingesetzt. Das ANFIS-Modell basiert auf einer hybriden Struktur, sodass es sowohl als ein Neuronales Netz, als auch ein Fuzzy-System interpretiert werden kann. In dem System sind Regeln angelegt, die nach dem TSK-Modell definiert sind (Takagi-Sugeno-Kang-Reglern 2.3.2). Die Abbildung (Abbildung hier zitieren) zeigt einen Model mit folgender Regelbasis:

$$R_1: \text{ Falls } x_1 \text{ ist } A_1 \text{ und } x_2 \text{ ist } B_2, \text{ dann ist } y=f_1(x_1,x_2)$$

$$\begin{aligned} R_2: & \text{ Falls } x_1 \text{ ist } A_1 \text{ und } x_2 \text{ ist } B_2, \text{ dann ist } y=f_2(x_1, x_2) \\ R_3: & \text{ Falls } x_1 \text{ ist } A_3 \text{ und } x_2 \text{ ist } B_3, \text{ dann ist } y=f_3(x_1, x_2) \end{aligned}$$

Dabei sind A_1, A_2, B_1 und B_2 linguistische Termen, die den entsprechenden Fuzzy-Mengen $\mu_i^{(j)}$ zugeordnet sind. Die Funktionen f_i sind linear und sehen wie folgt aus (siehe 2.3.3):

$$f_i(x_1, x_2) = p_0^i + p_1^i \cdot x_1 + p_2^i \cdot x_2 \quad (3.1)$$

Der Konklusionsfunktion f_i entspricht die Regel R_i . Somit hat jede Regel eine eindeutige Ausgangsfunktionen mit eindeutigen Parametern.

Die Ausgabe eines ANFIS-Modells berechnet sich genau so wie ein TSK-Modell (siehe 2.3.3). Für den genannten Beispiel lautet die Ausgabe:

$$f = \frac{\sum_i^3 \tilde{w}_i \cdot f_i(x_1, x_2)}{\sum_i^3 \tilde{w}_i} \quad (3.2)$$

Der ANFIS-Ansatz besteht aus 5 Schichten. Die Abbildung X veranschaulicht die Struktur.

In der **ersten Schicht** werden die Eingabewerte eingereicht und entsprechend die Zugehörigkeiten zu den Fuzzy-Sets ausgegeben. Weiterhin werden in der **zweiten Schicht** die Aktivierungswerte jeder Regel ausgewertet. Die Neuronen werden mit Π gekennzeichnet. Einzelnen Fuzzyzugehörigkeitswerte werden mittels Operatoren kombiniert, um die Aktivierungsgrad jeder Regel zu berechnen. Hier dürfen Operatoren zur Verknüpfung von Fuzzy-Mengen eingesetzt werden, üblicherweise der UND-Operator (siehe 2.2.1). Die Gleichung 3.3 bei UND-Verknüpfung gibt die Berechnung:

$$\tilde{w}_i = \prod \mu_i^j(x_j) \quad (3.3)$$

Die Funktion μ_i^j ist eine Zugehörigkeitsfunktion (referenz fehlt hier). Das \tilde{w}_i ergibt die Aktivierung, oder Erfüllungsgrad des Regels R_i .

Im **dritten Schicht** findet die Normalisierung aller Aktivierungswerte \tilde{w}_i statt. Es wird der Beitrag berechnet, den jeder Regel für den Gesamtausgabe beiträgt. Nach der Normalisierung erhält man Aktivierungsgrößen zwischen 0 und 1. Die Gleichung ?? gibt die normalisierten Werten aus:

$$\bar{w}_i = \frac{\tilde{w}_i}{\sum_j \tilde{w}_j} \quad (3.4)$$

Im **vierten Schicht** berechnen die mit N markierten Neuronen die gewichteten Ausgabewerten. Das "Gewicht" aus dem letzten Layer, bzw. Parameter, wird an der entsprechenden Ausgabefunktion multipliziert:

$$\bar{y}_i = net_i = \bar{w}_i \cdot f_i(x_1, \dots, x_n). \quad (3.5)$$

Im **fünften Schicht** steht ein einziges Neuron, der mit Σ beschriftet ist. Im letzten Schicht berechnet man die Ausgabe, indem alle Werte aus dem **vierten Schicht** zusammenaddiert werden:

$$y = y_{out} = \sum_i \bar{y}_i = \frac{\tilde{w}_i \cdot f_i(x_1, \dots, x_n)}{\sum_j \tilde{w}_j}. \quad (3.6)$$

Diese Struktur ist Äquivalent zu dem TSK-Modell. Für die Optimierung von Parameter der Zugehörigkeitsfunktionen und Konklusionsfunktionen eignet sich der ANFIS-Ansatz. Dieser Ansatz wurde ich auch bei dem Versuch, die Parameter eines Fuzzy-Modells zu optimieren.

Die Kleinste-Quadrate-Methode

3.1.2 Modelle mit verstärkendem Lernen

Bei Modellen mit verstärkendem Lernen, im Vergleich zu Modellen mit festen Lernaufgaben, wird versucht, die Menge der Daten für das Lernen möglichst gering zu halten. Weiter unterscheiden sich die beiden Modellen in dem, dass bei Modellen mit verstärkendem Lernen keine Vorwissen bekannt werden müssen, was öfters der Fall sein kann. Es reicht nur, wenn im Laufe des Lernens angegeben wird, ob die Richtung der Optimierung sinnvoll ist.

Ein Großes Problem beim verstärkendem Lernen besteht darin, hervorzusagen, wie groß der Einfluss einer Regelaktion auf das Gesamtsystem ist. Dieses Problem wird als *Credit Assignment Problem* bezeichnet.

Es existiert eine große Mengen von Modellen mit verstärkendem Lernen, alle aber basieren auf dem gleichen Prinzip. Das System wird zwei Teilsysteme aufgeteilt: zum einen der "Kritiker" (das "kritisierende System") und der Akteur (zuständig für die Anwendung und Abspeicherung der Regelungsstrategie). Der Kritiker äußert seine Meinung über den jetzigen Zustand unter Berücksichtigung der vorhergehenden Zustände und somit entscheidet der Akteur anhand diese Bewertung, ob eine Korrektur der Regelbasis gemacht wird.

Das NEFCON-Modell

Ziel des NEFCON-Modell ist es, eine interpretierbare Fuzz-Regelbasis mit möglichst kleine Trainingsschritte zu erlernen. Das Modell unterscheidet sich von dem ANFIS darin, dass es erlaubt einen Regelbasis, ohne Vorwissen zu erlernen. Dieses Modell bietet natürlich die Möglichkeit Vorwissen mitzubringen. Sowohl schon feste Lernaufgaben, Fuzzy-Systeme mit vorhandenen Regelbasis, als auch Systeme mit zum Teil gefüllter Fuzzy-Regelbasis. Das ist der größte Vorteil des NEFCON-Modell gegenüber anderen Modellen.

Das NEFCON-Modell basiert auf ein Mamdani-Regler. Zum Veranschaulichung wird hier einen kleinen Beispiel mit Grafik und Regelbasis gegeben.

$$\begin{aligned} R_1: & \text{ IF } x_1 \text{ in } A_1^{(1)} \text{ AND } x_2 \text{ in } A_1^{(2)}, \text{ THEN } y \text{ is } B_1 \\ R_2: & \text{ IF } x_1 \text{ in } A_1^{(1)} \text{ AND } x_2 \text{ in } A_2^{(2)}, \text{ THEN } y \text{ is } B_1 \\ R_3: & \text{ IF } x_1 \text{ in } A_2^{(1)} \text{ AND } x_2 \text{ in } A_2^{(2)}, \text{ THEN } y \text{ is } B_2 \\ R_4: & \text{ IF } x_1 \text{ in } A_3^{(1)} \text{ AND } x_2 \text{ in } A_2^{(2)}, \text{ THEN } y \text{ is } B_3 \\ R_4: & \text{ IF } x_1 \text{ in } A_3^{(1)} \text{ AND } x_2 \text{ in } A_3^{(2)}, \text{ THEN } y \text{ is } B_3 \end{aligned}$$

Die NEFCON-Modell basiert auf einem generischen Fuzzy-Perzeptron. Das Modell konnte in drei Schichten aufgeteilt werden. Erste Schicht besteht natürlich aus den Eingangsneuronen. Die ist eindeutig und will nicht weiter darauf eingehen. In der zweiten Schicht befinden sich die inneren Neuronen. Diese spielen aus unserem Fuzzy-System die Regeln. Im Beispiel sind insgesamt fünf Regeln gegeben. Die Verbindungen zum Regeln fließen aus den entsprechenden Eingangsknoten. Die Fuzzy-Mengen $\mu_r^{(i)}$, die in Mehrere Regeln wirken, werden durch Ellipse gekennzeichnet. Falls dann beim Lernen eine Anpassung an diese Gewicht geführt werden soll, muss dies in allen gleichen Verbindungen gemacht werden.

Das eigentliche Lernprozess besteht aus zwei Phasen. In der erste Phase wird versucht eine Regelbasis zu erlernen. Diese Phase wurde weggelassen, falls schon eine mitgeliefert wird. Es lässt sich auch unvollständige Regelbasis vervollständigen. Natürlich das Erlernen einer Regelbasis dauert am längsten von den drei Fällen. In der zweite findet die Optimierung statt. Dabei werden Fuzzy-Sets modifiziert oder selbst die Verbindungen zu Regeln umgetauscht [<http://www.witi.cs.uni-magdeburg.de/~nuernb/wsc2/>]. Wie oben schon erwähnt wurde, wird ein Kritiker gebraucht. In dem NEFCON-Modell wird als Bewertungsmaß ein Fuzzy-Error verwendet. Damit die Optimierung optimal ausgeführt werden kann, sollte das Vorzeichen der Ausgabewert bekannt sein. Darüber hinaus wird ein erweiterter Fuzzy-Error E^* berechnet:

$$E^* = \text{sgn}(y_{out}) \cdot E(x_1, \dots, x_n) \quad (3.7)$$

Erlernen einer Regelbasis

Es existieren mehrere Algorithmen zum Erlernen von Regelbasis. Methoden können in drei Kategorien aufgeteilt werden: Methoden, die ohne vordefinierte Regelbasis startet; solche, die mit alle möglichen Regeln in dem Regelbasis starten und entsprechend solche mit unvollständige Basis. In den folgenden zwei Unterkapitel werden Methode für die ersten zwei Kategorien vorgestellt.

Top-Down- oder Reduktionsmethode zum Erlernen einer Regelbasis

Zum einen steht die Methode des Top-Down-Methode (in der Literatur auch als NEFCON I bekannt) als ein Verfahren zur Erlernen von Regelbasen. Diese Methode startet mit einer Regelbasis, die aus allen möglichen Regeln besteht. Diese Basis entsteht aus alle Kombinationen von Ein-/Ausgabewerten in Regeln. Daraus entstehen auch widersprüchliche Regeln.

Wenn die Regelbasis zur Verfügung steht, liegt diese einer Analyse unter. Das Prozess kann in zwei Phasen aufgeteilt. Im ersten Teil werden Regeln, die eine Ausgabe mit falschen Vorzeichen liefert (siehe die erweiterte Fuzzy-Error 3.7). In der zweiten Phase werden Regeln der zufällig ausgewählt. Folglich wird für jede Regel der Fehler berechnet. Die Regeln mit identischer Prämisse werden gruppiert. Am Ende wird die Regel aus der Gruppe, die den kleinsten Fehlerrate hat. Die restlichen Regeln aus der Gruppe werden verworfen.

Nachteil des Top-Down-Methode besteht darin, dass es mit sehr großen Regelbasis startet und somit sehr Aufwändig ist. Die nächste Methode ist das Gegenteil von Top-Down-Methode, zwar Bottom-Up-Methode.

Bottom-Up- oder Eliminationsmethode zum Erlernen einer Regelbasis

Der Bottom-Up-Algorithmus beginnt mit einer leeren Regelbasis. Jedoch muss eine initiale Aufteilung(Intervall) der Ein- und Ausgabewerten gegeben sein. Wie der Top-Down besteht diese Methode auch aus zwei Phasen.

Erste Phase beginnt mit der Bestimmung der Prämisse für die Regeln. Dies wird gemacht, indem die Eingangsgrößen zu den Eingabewerten ausgewertet werden. Aus den Fuzzy-Mengen (Eingangsgrößen), die den größten Zugehörigkeitsgrad bei bestimmter Eingabe liefern, wurden Prämisse für eine Regel gebaut. Danach versucht das Algorithmus eine geeignete Ausgabe aus dem aktuellen Fuzzy-Fehler zu "raten". Damit das Raten funktioniert, wird angenommen, dass Eingaben mit ähnlichen Fehlerwerten ähnliche Ausgaben liefern.

In der zweiten Phase werden die Fuzzy-Mengen in den Konklusionen optimiert. Das Prozess versucht, falls nötig ist, die Fuzzy-Menge einer Konklusion durch eine andere zu ersetzen, anstatt die Parameter in der Konklusion zu optimieren.

Wegen des inkrementellen Lernen lässt sich einfach Vorwissen in dem Regelbasis integrieren. In so einem Fall werden fehlende Regeln hinzugefügt. Die Regelbasis kann aufgebaut, bzw. erlernt werden, aber es ist keine Garantie das die Basis überhaupt passend ist, oder immer eine Regelbasis gefunden wird. Aus diesem Grund ist es empfohlen, dass die Regelbasis noch manuell am Ende des Algorithmus überprüft und entsprechend angepasst wird.

Optimierung der Regelbasis

Die Optimierung bei NEFCON verwendet die Methode "Back Propagation Method", oder Rückpropagationsmethode. Der Fehler wird rückwirkend durch das Netz geführt und lokal bei jeder Fuzzy-Menge angewendet.

Eine Änderung darf sowohl in den Prämissen, als auch in den Konklusionen. Es wird das Prinzip des verstärkenden Lernens angewendet. Das bedeutet, dass für jede Änderung eine Fuzzy-Menge entweder "bestraft" oder "belohnt" wird. Bestrafung und Belohnung werde in Änderungen wie Verschiebung, Vergrößerung oder Verkleinerung des Bereichs einer Fuzzy-Menge ausgedrückt. Änderungen werden entsprechend iterative in Beziehung zum Fuzzy-Fehler gemacht.

Schlussworte

[Computational Intelligence]

Literaturverzeichnis

- AM01. ANDREA TETTAMANZI und MARCO TOMASSINI: *Soft Computing: Integrating Evolutionary, Neural, and Fuzzy Systems*. Springer-Verlag Berlin Heidelberg, 2001.
- Han98. HANS-HEINRICH BOTHE: *Neuro-Fuzzy-Methoden*. Springer-Verlag Berlin Heidelberg, 1998.
- JCE97. JYH-SHING ROGER JANG, CHUEN-TSAI SUN und EIJI MIZUTANI: *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Prentice Hall, 1997.
- RCC⁺15. RUDOLF KRUSE, CHRISTIAN BORGELT, CHRISTIAN BRAUNE, FRANK KLAWONN, CHRISTIAN MOEWES und MATTHIAS STEINBRECHER: *Computational Intelligence: Eine methodische Einführung in Künstliche Neuronale Netze, Evolutionäre Algorithmen, Fuzzy-Systeme und Bayes-Netze*. Springer Fachmedien Wiesbaden, 2015.

A

Glossar

DisASter	DisASter (Distributed Algorithms Simulation Terrain), A platform for the Implementation of Distributed Algorithms
DSM	Distributed Shared Memory
AC	Linearisierbarkeit (atomic consistency)
SC	Sequentielle Konsistenz (sequential consistency)
WC	Schwache Konsistenz (weak consistency)
RC	Freigabekonsistenz (release consistency)