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

Das Thema dieser Ausarbeitung ist Unschärfe Informationen, oder Fuzzy-Sets. Mit Fuzzy-Sets lassen sich schwammige Daten, wie "großeSZahlen 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 verwenden wird, Neuro-Fuzzy-Systeme aufzubauen.

2

Unscharfe Information und Fuzzy-Sets

Man ist gewohnt in der Mathematik, Informatik und andere genaue Wissenschaften immer von Genauigkeit zu reden. Jedoch gibt es viele Aussagen die nicht entweder wahr oder falsch sind. Manche haben vielleicht nichts damit zu tun. Aber Andere, die nicht mit der klassische Logik (wahr oder falsch, 0 oder 1) definiert werden können, sind sehr interessant. Das wesentliche Ziel der Fuzzy-Logik ist es die Mangeln, die die klassische Logik hat zu überwinden.

2.1 Fuzzy-Sets

Es ist kein Wunder, dass die Fuzzy-Theorie, wie sie damals in dem Jahr 1996 veröffentlicht wurde, keinen Platz unter den genauen Wissenschaften finden konnte. Heutzutage jedoch spielt die Theorie eine fundamentale Rolle bei Analyse und Vorhersage von Ereignisse. Viele Spezialisten verwenden unscharfe Informationen und versuchen mit der Fuzzy-Logik natürliche Katastrophen, oder Krisen hervorzusagen.

Damit unscharfe Daten von Maschinen überhaupt verarbeitet werden können, müssen sie zunächst in entsprechender Darstellung gebracht werden. Ein Vorgehen wird aus der klassischen Logik geklaut. Man nimmt die Zweiwertigkeitsprinzip:

$$\text{nein} \leftrightarrow \text{ja}$$
$$\text{falsch} \leftrightarrow \text{wahr}$$
$$0 \leftrightarrow 1$$

und erweitert die binäre Zugehörigkeit, dass ein Element entweder vollständig oder gar nicht dazugehört. Man gibt statt dessen Elemente eine graduelle Zugehörigkeitsgrad. Also ein Element gehört weniger zu einer Menge als ein Anderes. in dem Kontext würden wir die oben definierten Antworten in:

von "gehört nicht zut Menge“ bis "gehört zur Menge“
von nein bis ja
von falsch bis wahr
von 0 bis 1

umwandeln.

Zum Verdeutlichen betrachten wir die Menge M der reellen Zahlen, die viel größer sind als 1 [Fuzzy-Logik und Fuzzy-Control, Jörg Kahlert; Hubert Frank].

$$M = \{x \mid x \in \mathfrak{R},\ x >> 1\} \tag{2.1}$$

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

Aus der Graphik kann man feststellen, das 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, 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. Da sieht man sogar, dass 10 selbst mit einer Zugehörigkeit von 0.15 ist, und erst bei Werte größer als 20 liefert die Funktion $\mu(x)$ eine 1 (volle Zugehörigkeit). In solchen Fällen und Probleme eignet sich sehr gut Fuzzy-Sets.

Für die Zugehörigkeit von Elementen zu einer Mengen wird meistens Werte zwischen 0 und 1. Dies wird auch in dieser Ausarbeitung fest gesetzt, da Zahlen leichter zu Vergleichen sind und auf dieser Weise kann man deutlich Sehen welche Zahlen "besserßu einer Menge gehören.

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 klassische Logik gut. Im folgenden Kapitel werden die einzelnen Operationen beschrieben[Computational Intelligence].

2.2.1 Durchscnitt

In der klassische 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:

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

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

Der Durchschnitt der beiden Mengen ergibt sich:

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

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 methematischer 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:*

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

*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:*

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

*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 großer Rolle [Fuzzy Logik und Fuzzy-Control Jörg Kahlert, Hubert Frank].

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: $(\mu_1 \oplus \mu_2) = \mu_1(x) + \mu_2(\mathrm{x})$ - $\mu_1(x) \cdot \mu_2(x)$
3. abgeschnittene Differenz: $(\mu_1 \dot{-} \mu_2) = \mathrm{MAX}(0, \mu_1(\mathrm{x}) + \mu_2(\mathrm{x}) - 1)$
4. abgeschnittene Summe: $(\mu_1 \hat{+} \mu_2) = \mathrm{MIN}(1, \mu_1(\mathrm{x}) + \mu_2(\mathrm{x}))$
5. $(\mu_1 \dot{-} \mu_2) = \mathrm{MIN}(\mu_1(\mathrm{x}), 1 - \mu_2(\mathrm{x}))$
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:*

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

*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. Dies ist eine Vorbereitung auf die weiteren Kapitel. In den nächsten Kapiteln beschäftige ich mich mit der Möglichkeit mehrere Fuzzy-Mengen so zu kombinieren, dass man spezifisch eine Aussage trifft. Das würde heißen, dass wir mehrere Fuzzy-Mengen analysieren werden und daraus bestimmte Schlüsse ziehen wollen. Ein Beispiel dafür wäre: Wenn die Tomate rot ist, dann ist die reif. Folgende Logische Aussagen können sehr einfach mit Fuzzy-Logik beschrieben werden. In der Literatur taucht die Name Fuzzy-Regeln. Diese können dann zusammengestellt werden, um Fuzzy-Regel-Systeme aufzubauen. In dem nächsten Kapitel wird Fuzzy-Regel-Systeme vorgestellt.

2.3 Fuzzy-Regelsysteme

Fuzzy-Regelsysteme sind solche Systeme, die unscharfe Informationen als Eingabe bekommen und eine unscharfe Ausgabe liefern. Diese Systeme sind sehr einfach zu verstehen und definieren. Die basieren auf einfache IF-THEN Klauseln. In diesem Kapitel werde ich zwei Arten von Reglern.

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:

$$R : \text{If } x_1 \text{ is } \mu_R^{(1)} \text{ and } ... \text{ and } x_n \text{ is } \mu_R^n \text{ then } y \text{ is } \mu_R \tag{2.5}$$

In der Regel sind x_1, ..., x_n die Eingangsgrößen, $\mu_R^{(n)}$ die Fuzzy-Sets, bzw. linguistische Werte(üngefähr null", "mittelgroßüsw.), der Prämisse und entsprechend μ_R der Konklusion.

Mamdani-Reglers können ziemlich einfach verstanden werden. Der Regler ist eine Funktion, die eine endliche Menge von Ausgaben(Konklusion) bei einer endlichen Menge von Eingabewerten liefert. Man kann somit eine Funktion definiert die folgender Form besitzt:

$$f(x_1, ..., x_n) \approx \begin{cases} \mu_{R_1} & \text{falls } x_1 \approx \mu_{R_1}^{(1)} \text{ und } ... \text{ und } x_n \approx \mu_{R_1}^{(n)} \\ \vdots & \\ \mu_{R_r} & \text{falls } x_1 \approx \mu_{R_r}^{(1)} \text{ und } ... \text{ und } x_n \approx \mu_{R_r}^{(n)} \end{cases} \tag{2.6}$$

Hier ist ziemlich klar zu verstehen, dass die Funktion genau so viele Ausgaben hat, wie es Regeln gibt. Oder genau r Ausgaben. Es ist noch zu erwähnen, dass die Funktionswerte eigentlich eine Zugehörigkeit zur Fuzzy-Menge sind. In dieser Form kann man sehr einfach das Regler lesen.

Mit einem Mamdani-Regler können viele Probleme aus der reellen Welt definiert werden. Als kleines Beispiel können wir Tomaten nehmen. Ich verwende die

Farben einer Tomate als Eingabewerte. Die Ausgabe würde uns sagen, ob eine Tomate reif 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. Nehmen wir an, dass es möglich wäre, die Farbe als Wert aus einer Tomate auslesen zu können. Somit für jeder Wert könnte die Zugehörigkeit für jede Menge ausgerechnet werden. Unsere Regelbasis würde dann wie folgt aussehen:

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.

Wenn man nur die drei Farben betrachten würde, in dem ersten Augenblick bekommt recht einfache if-then-Ausdrücke. Es ist aber bekannt, dass Beispiel aus der Praxis viel komplizierter sind, z.B. Fuzzy-System zur hervorsage von dem Wetter, zur Erkennung von Stromdiebstahl usw.

Eine Tabelle für die Fuzzy-Relationen ist unten gegeben

x $\diagdown$ 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:

x $\diagdown$ 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

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 jedoch steigt die Komplexität eines Systems proportional.

2.3.2 Takagi-Sugeno-Kang-Regler

Takagi-Sugeno-Kang-Regler erweitern die oben vorgestellte Modell der Mamdani-Reglar. TSK-Regler verwenden Regeln der Form:

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

Genause wie bei dem Mamdani-Regler beschreiben die Eingabewerte unscharfe Informationen. Der Unterschied ergibt sich in der Konklusion. In der Konklusion steht eine Funktion, anstatt eine Fuzzy-Menge. Dies ergibt die Möglichkeit eine Regelbasis so zu definieren, dass es für jede Regel eine Separate Funktion berechnet wird.

TSK-Reglern funktionieren genau so wie Mamdani-Reglern. Für eine spezifische Eingabe wird der Zugehörigkeitswert zum Fuzzy-Mengen berechnet. In jedem Regel wird dies geprüft und festgestellt, welche Prämisse am vorausgesetzt ist und die zugehörige Funktion wird berechnet.

TSK-Regler, da die sehr viel einer Mamdani ähneln, können im Spezialfall Mamdani-Regler repräsentieren. Man definiert jede Funktion f_R konstant. Als Beispiel sind folgende Regeln gegeben:

$$R_1: \text{ If } x \text{ is ´sehr klein´ then } y = xR_2: \text{ If } x \text{ is ´klein´ then } y = 1R_3: \text{ If } x \text{ is ´groß´ then } y = x - 2R_4 \tag{2.8}$$

Zunächst wird für jede Fuzzy-Set (´sehr klein´, ´klein´, ´groß´, ´sehr groß´) einen Bereich definiert. Die Bereiche jeder Menge können auch überlappen. Anhängig davon ergibt sich eine unterschiedliche Funktionskurve für y. Zur Veranschaulichung werden sechs Grafiken vorgezeigt.

2.3.3 Fuzzifizierung und Defuzzifizierung

3

Neuro-Fuzzy-Systeme

3.1 Neuro-Fuzzy-Regler

3.1.1 Modelle für feste Lernaufgaben

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 dann eine Verarbeitung, die als Ziel eine Optimierung hat. Dafür sind entsprechend auch die Ein-/Ausgabedaten gebraucht.

Ein weiteres Anwendungsbeispiel von solchen Modellen ist, wenn eine bereits existierende Regelbasis mit einem neuen Fuzzy-Regelbasis ausgetauscht werden soll. Falls die Basis schon errechnete Werte geliefert hat, können diese Zusammen mit den zugehörigen Eingabewerten dem Neuro-Fuzzy-System gegeben werden. Am Ende erhaltet man einen Optimierten Fuzzy-Regel-Basis, die die alte ërsetzt".

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. Eine solcher Verfahren wird später noch vorgestellt.

Folglich wird einen Beispiel von einem Modell für feste Lernaufgaben vorgestellt. Es geht nämlich um den 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 wurde das Modell in mehrere Software-Pakete schon eingesetzt. Das ANFIS-Modell basiert auf einer hybriden Struktur, sodass es von sowohl einen Neuronales Netz als auch von einem Fuzzy-System interpretiert werden kann. In dem System sind Regeln angelegt, die nach dem TSK-Modell definiert sind (Takagi-Sugeno-Kang-Reglern). Auf dem Grafik ist ein Beispiel eines ANFIS-Modell mit folgenden drei Regeln:

R1: Falls x_1 ist A_1 und x_2 ist B_2, dann ist $y{=}f_1(x_1{,}x_2)$

R2: Falls x_1 ist A_1 und x_2 ist B_2, dann ist y=$f_2(x_1,x_2)$
R2: Falls x_1 ist A_3 und x_2 ist B_3, dann ist y=$f_3(x_1,x_2)$

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:

$$f(x_1, x_2) = p_i \cdot x_1 + q_i \cdot x_2 + r_i \tag{3.1}$$

Der ANFIS-Ansatz besteht aus 5 Schichten.

In der **ersten Schicht** werden die Eingabewerte eingereicht und entsprechend die Zugehörigkeiten zu den Fuzzy-Sets ausgegeben. Weiterhin in der **zweiten Schicht** werden die Prämisse aus den Regeln ausgewertet. Hier dürfen mehrere Operatoren zur Verknüpfung von Fuzzy-Mengen eingesetzt werden, z.B. der UND-Operator (siehe 2.2.1). In diesem Fall wird zur Auswertung der Prämisse das Produkt-t-Norm Formel verwendet. Dafür steht folgende Formel zur Berechnung:

$$\tilde{a}_i = \prod \mu_i^j(x_j) \tag{3.2}$$

Das $\tilde{a}_i$ ergibt die Aktivierung, oder Erfüllungsgrad des Regels R_i.

Im **dritten Schicht** findet die Normalisierung aller Aktivierungswerte $\tilde{a}_i$ statt. In Einfachen Worten: es wird der Beitrag berechnet, den jeder Regel für den Gesamtausgabe beiträgt. Die Zugehörigkeitswerte $\tilde{a}_i$ bekommen nach der Normalisierung einen Wert zwischen 0 und 1. Dies wird entsprechend mit folgende Formel berechnet:

$$\bar{a}_i = \frac{\tilde{a}_i}{\sum_j \tilde{a}_j} \tag{3.3}$$

Im **vierten Schicht** berechnen die Neuronen die gewichteten Ausgabewerten der entsprechenden Eingabewerten. Das "Gewicht", bzw. Parameter, wird an die entsprechende Funktion multipliziert. Das folgt die Formel:

$$\bar{y}_i = net_i = \bar{a}_i \cdot f_i(x_1, ..., x_n) \tag{3.4}$$

Die Neuronen im **fünften Schicht** berechnen die Gesamtausgabe. Die Ausgabe wird aus die Summe aller gewichteten Ausgabewerten der einzelnen Regeln. Mit eine Umschreibung erhalten wir dann:

$$y = a_{out} = net_{out} = \sum_i \bar{y}_i = \frac{\tilde{a}_i \cdot f_i(x_1, ..., x_n)}{\sum_j \tilde{a}_j} \tag{3.5}$$

Das ANFIS-Modell ermöglicht die Optimierung von Modellparametern, d.h. Die Fuzzy-Mengen- und die Ausgabefunktionsparametern. Diese können erlernt werden, wenn eine angemessene Lernaufgabe vorliegt. Außerdem muss eine ausreichende Menge von Ein-/Ausgabe-Werten zur Verfügung stehen.

Es existieren mehrere Lernmethoden zur Optimierung der Parametern. Zwei davon sind Gradientenverfahren(analog zur Fehler-Rückpropagation der neuronalen Netze) und die Kleinste-Quadrate-Methode. Den zweiten Modell werde ich im nächsten Unterkapitel vorstellen.

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 "kritisierendeSSystem) und der Aktor(zuständig für die Anwendung und Abspeicherung der Regelungsstrategie). Der Kritiker "äußertßeine Meinung über den jetzigen Zustand unter Berücksichtigung der vorhergehenden Zustände und somit entscheidet der Aktor 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.

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

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 Eingangknoten. 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://wwwiti.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:

$$E^* = sgn(y_{out}) \cdot E(x_1, \ \ldots, \ x_n) \tag{3.6}$$

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 3.6). 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 Propagtion 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 "bestraftöder "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

CDK02. Coulouris, George, Jean Dollimore und Tim Kindberg: *Verteilte Systeme: Konzepte und Design*. Addison-Wesley-Verlag, 2002.

Che85. Cheriton, David R.: *Preliminary Thoughts on Problem-oriented Shared Memory: A Decentralized Approach to Distributed Systems*, 1985.

Mal97. Malte, Peter: *Replikation in Mobil Computing*. Seminar No 31/1997, Institut für Telematik der Universität Karlsruhe, Karsruhe, 1997. `http://www.ubka.uni-karlsruhe.de/cgi-bin/psview?document=/ira/1997/31`.

Mos93. Mosberger, David: *Memory Consistency Models*. Technical Report 93/11, University of Arizona, November 1993.

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)