5 Das NEFCON-Model

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

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

5.1 Erlernen einer Regelbasis

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

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

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

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

5.1.1 Beispiel des erlernens einer Regelbasis

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

Die durch dieses Lernverfahren entstandenen Regeln sind wiefolgt:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

5.2 Erlernen von Fuzzy-Mengen

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

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

Das Lernverfahren selbst besteht aus sechs Einzelschritten:

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

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

Schritt 3 Bestimme das Vorrücken des Stellwertes im neuen Systemzustand.

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

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

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

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

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

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

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

5.3 Nachteile des NEFCON-Models

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

Literatur

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

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

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

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

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

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

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

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