Ein Neuro–Fuzzy System zur Funktionsapproximation

Detlef Nauck
Institut für Informations– und Kommunikationssysteme
Otto–von–Guericke–Universität Magdeburg
Universitätsplatz 2, D–39106 Magdeburg
Tel.: (0391) 67-12700 Fax: (0391) 67-12018, E-Mail: nauck@iik.cs.uni-magdeburg.de

Zusammenfassung

In diesem Beitrag wird eine Neuro–Fuzzy–Architektur zur Funktionsapproximation vorgestellt. Der Lernalgorithmus ist in der Lage, sowohl die Struktur als auch die Parameter eines Fuzzy–Systems zu bestimmen. Der Ansatz entspricht einer Erweiterung der bereits vorgestellten Modelle NEFCON und NEFCLASS, die für regelungstechnische Anwendungen bzw. Klassifikationsaufgaben eingesetzt werden. Das vorgestellte Modell NEFPROX ist allgemeiner und kann für beliebige auf Funktionsapproximation basierenden Anwendungen eingesetzt werden.

1 Einführung

Bestimmte Fuzzy–Systeme sind wie bestimmte Neuronale Netze auch universelle Funktionsapproximatoren [1, 4]. Um für ein gegebenes Problem ein geeignetes Fuzzy–System zu finden, müssen Zugehörigkeitsfunktionen (Parameter) und eine Regelbasis (Struktur) spezifiziert werden [5]. Dies kann entweder durch Vorwissen, einen Lernalgorithmus oder durch beides geschehen. Falls ein Lernverfahren eingesetzt wird, das nur lokale Informationen verwendet und auch nur lokale Änderungen vornimmt, spricht man meist von einem Neuro-Fuzzy–System [9].

Moderne Neuro–Fuzzy–Systeme [7] werden meist als mehrschichtiges, vorwärtsbetriebenes Neuronales Netz dargestellt, wobei sich jedoch die Verbindungsgewichte und die in den Neuronen enthaltenen Funktionen von herkömmlichen Neuronalen Netzen unterscheiden. Ein solches Neuro–Fuzzy–System zeichnet sich durch die folgenden Eigenschaften aus:

(i) Es besteht aus einem Fuzzy–System, das mit Hilfe eines aus dem Bereich der Neuronalen Netze stammenden Lernalgorithmus trainiert wird. Das (heuristische) Lernverfahren operiert auf der Basis lokaler Information und führt auch nur lokale Änderungen im Fuzzy–System durch.

(ii) Es läßt sich als dreischichtiges, vorwärtsbetriebenes Neuronales Netz darstellen, wobei die erste Schicht die Eingaben, die mittlere Schicht die Fuzzy–Regeln und die dritte Schicht die Ausgaben repräsentieren. Die Fuzzy–Mengen sind als Verbindungsgewichte kodiert. Davon abweichend wird manchmal eine fünfschichtige Struktur gewählt, wobei die Fuzzy–Mengen in den Neuronen der zweiten und vierten Schicht kodiert werden.

(iii) Es ist jederzeit als ein System aus Fuzzy–Regeln interpretierbar.

(iv) Das Lernverfahren berücksichtigt die semantischen Eigenschaften des zugrundeliegenden Fuzzy–Systems, was zu einer Einschränkung zulässiger Parametermodifikatonen führt.

In [6] bzw. [8] wurden die Ansätze NEFCON und NEFCLASS vorgestellt. NEFCON wird mit verstärkendem Lernen trainiert und für regelungstechnische Problemstellungen eingesetzt, während NEFCLASS ein gewöhnliches überwachtes Lernverfahren nutzt und zur Klassifikation von Daten dient. Beide Modelle sind zum Struktur– und Parameterlernen in der Lage, indem sie Varianten von „Fuzzy–Error–Backpropagation“ einsetzen. Die Bezeichnung „Backpropagation“ deutet hier an, daß der Lernvorgang auf dem Zurückpropagieren von Fehlersignalen durch die Systemarchitektur beruht, wobei lokale Parametermodifikationen durchgeführt werden. Es handelt sich hierbei nicht um ein Gradientenverfahren wie es in Neuronalen Netzen zur Anwendung kommt, sondern um eine einfache Heuristik, da die im Neuro–Fuzzy–System verwendeten Funktionen gewöhnlich nicht differenzierbar sind.

Der Begriff „Fuzzy–Fehler“ weist darauf hin, daß das Fehlermaß zur Steuerung des Lernvorgangs entweder durch eine Fuzzy–Regelbasis (NEFCON) oder durch eine Fuzzy–Menge über den Differenzen zwischen erwünschter und tatsächlicher Ausgabe modelliert wird. In beiden Fällen ist der Lernvorgang überwacht. Falls der Fehler durch eine Regelbasis zur Beschreibung von Fehlerzuständen angegeben wird, so handelt es sich um eine spezielle Art überwachten Lernens – das verstärkende Lernen (reinforcement learning). Ist die erwünschte Ausgabe jedoch bekannt, so handelt es sich um einfaches überwachtes Lernen.

In diesem Beitrag wird eine Methode zur Funktionsapproximation durch ein Neuro–Fuzzy–Modell vorgestellt.

2 Die Modellarchitektur

Funktionsapproximation auf der Grundlage lokaler Lernstrategien ist eines der bedeutendsten Anwendungsgebiete Neuronale Netze [2] und Neuronaler Fuzzy–Systeme [9]. Neuro–Fuzzy-Systeme haben den Vorteil, A–priori–Wissen nutzen zu können, während Neuronale Netze ohne Vorwissen auskommen müssen (learning from scratch) und aufgrund ihrer verteilten Informationsspeicherung einen Black–Box–Charakter aufweisen, der eine Interpretation in Form von Regeln meist verhindert.

Ein bekanntes Neuro–Fuzzy–System zur Funktionsapproximation ist das ANFIS–Modell [3]. Für dieses Modell ist jedoch kein Algorithmus zum Strukturlernen angegeben, und es implementiert ausschließlich Sugeno–Fuzzy–Modelle, die nur differenzierbare Funktionen verwenden (z.B. Produkt als t–Norm).

Wir schlagen ein allgemeineres Modell vor, das wie NEFCON und NEFCLASS auf einem generischen Fuzzy–Perceptron beruht [9].

Definition 1 *Ein dreischichtiges Fuzzy–Perceptron ist ein dreischichtiges, vorwärtsbetriebenes Neuronales Netz* $(U, W, \mathrm{NET}, A, O, \mathrm{ex})$*, mit den folgenden Spezifikationen:*

(i) $U = \bigcup_{i \in M} U_i$ *ist eine nicht–leere Menge von Verarbeitungseinheiten, und* $M = \{1, 2, 3\}$ *heißt Indexmenge von* U*. Es gilt für alle* $i, j \in M, U_i \neq \emptyset$ *und* $U_i \cap U_j = \emptyset$ *für* $i \neq j$*.* U_1 *heißt Eingabeschicht,* U_2 *Regelschicht und* U_3 *Ausgabeschicht.*

(ii) Die Netzwerkstruktur ist $W : U \times U \to \mathcal{F}(\mathbb{R})$*, wobei ausschließlich Verbindungen* $W(u, v)$ *mit* $u \in U_i,\ v \in U_{i+1}\ (i \in \{1, 2\})$ *existieren.*

(iii) A *ordnet jeder Einheit* $u \in U$ *eine Aktivierungsfunktion* A_u *zur Berechnung der Aktivierung* a_u *zu. Es gilt*

(a) für Eingabeeinheiten und Regeleinheiten $u \in U_1 \cup U_2$*:*

$$A_u : \mathbb{R} \to \mathbb{R}, \quad a_u = A_u(\mathrm{net}_u) = \mathrm{net}_u,$$

(b) für Ausgabeeinheiten $u \in U_3$*:*

$$A_u : \mathcal{F}(\mathbb{R}) \to \mathcal{F}(\mathbb{R}), \quad a_u = A_u(\mathrm{net}_u) = \mathrm{net}_u.$$

(iv) O *ordnet jeder Einheit* $u \in U$ *eine Ausgabefunktion* O_u *zur Berechnung der Ausgabe* o_u *zu. Es gilt*

(a) für Eingabeeinheiten und Regeleinheiten $u \in U_1 \cup U_2$*:*

$$O_u : \mathbb{R} \to \mathbb{R}, \quad o_u = O_u(a_u) = a_u,$$

(b) für Ausgabeeinheiten $u \in U_3$*:*

$$O_u : \mathcal{F}(\mathbb{R}) \to \mathbb{R}, \quad o_u = O_u(\mathrm{net}_u) = \mathsf{DEFUZZ}_u(\mathrm{net}_u),$$

dabei ist DEFUZZ_u *ein geeignetes Defuzzifizierungsverfahren.*

(v) NET *ordnet jeder Einheit* $u \in U$ *eine Netzeingabefunktion* NET_u *zur Berechnung der Netzeingabe* net_u *zu. Es gilt*

(a) für Eingabeeinheiten $u \in U_1$*:*

$$\mathrm{NET}_u : \mathbb{R} \to \mathbb{R}, \quad \mathrm{net}_u = \mathrm{ex}_u,$$

(b) *für innere Einheiten $u \in U_2$:*

$$\mathrm{NET}_u : (\mathbb{R} \times \mathcal{F}(\mathbb{R}))^{U_1} \to [0,1], \;\; \mathrm{net}_u = \mathop{\top}_{u' \in U_1} (W(u',u)(o_{u'})),$$

dabei ist $\top$ eine t–Norm,

(c) *für Ausgabeeinheiten $u \in U_3$:*

$$\begin{aligned} \mathrm{NET}_u &: ([0,1] \times \mathcal{F}(\mathbb{R}))^{U_2} \to \mathcal{F}(\mathbb{R}), \\ \mathrm{net}_u &: \mathbb{R} \to [0,1], \;\; \mathrm{net}_u(x) = \mathop{\perp}_{u' \in U_2} (\top(o_{u'}, W(u',u)(x))), \end{aligned}$$

dabei ist $\perp$ eine t–Conorm.
Falls die Fuzzy–Mengen $W(u',u),\ u' \in U_2,\ u \in U_3$, über ihren Trägern monoton sind, und $W^{-1}(u',u)(\tau)$ das $x \in \mathbb{R}$ bestimmt, für das $W(u',u)(x) = \tau$ gilt, so kann alternativ für eine Ausgabeeinheit $u \in U_3$ gelten:

$$\mathrm{net}_u(x) = \begin{cases} 1 & \textit{falls } x = \dfrac{\sum\limits_{u' \in U_2} o_{u'} \cdot W^{-1}(u',u)(o_{u'})}{\sum\limits_{u' \in U_2} o_{u'}} \\ 0 & \textit{sonst.} \end{cases}$$

Für die Ausgabe o_u gilt dann abweichend von (iv.b):

$$o_u = x, \;\; \textit{mit}\, \mathrm{net}_u(x) = 1.$$

(vi) $\mathrm{ex} : U_1 \to \mathbb{R}$, *ordnet jeder Eingabeeinheit $u \in U_1$ ihre externe Eingabe $\mathrm{ex}(u) = \mathrm{ex}_u$ zu. Für andere Einheiten ist ex nicht definiert.*

Ein generisches Fuzzy–Perceptron kann als dreischichtiges Neuronales Netz interpretiert werden, das spezielle Aktivierungs– und Propagationsfunktionen verwendet und seine Gewichte in Form von Fuzzy–Mengen repräsentiert. Es entspricht andererseits aber auch einem Fuzzy–System, das in einer neuronalen Architektur repräsentiert wird. Um ein NEFCON– oder NEFCLASS–System aus einem generischen Fuzzy–Perceptron abzuleiten, sind gewisse Restriktionen anzugeben. Die Einschränkungen für ein Neuro–Fuzzy–Modell zur Funktionsapproximation entsprechen in etwa denen des NEFCON–Modells.

Definition 2 *Ein NEFPROX–System (Neuro–Fuzzy–Funktionsapproximation) ist ein dreischichtiges Fuzzy–Perceptron mit den folgenden Spezifikationen:*

(i) *Die Eingabeeinheiten werden mit $x_1, \ldots, x_n$, die inneren Regeleinheiten mit $R_1, \ldots, R_k$, und die Ausgabeeinheiten mit $y_1, \ldots, y_m$ bezeichnet.*

(ii) *Jede Verbindung zwischen Einheiten x_i und R_r ist mit einem linguistischen Term $A^{(i)}_{k_r}$ $(k_r \in \{1, \ldots, p_i\})$ benannt.*

(iii) *Jede Verbindung zwischen Einheiten R_r und einer Ausgabeeinheit y_j ist mit einem linguistischen Term $B_{k_r}^{(j)}$ $(k_r \in \{1, \ldots, q_j\})$ benannt.*

(iv) *Verbindungen, die von derselben Eingabeeinheit x_i wegführen und identische Bezeichnungen tragen, besitzen zu jeder Zeit dasselbe Fuzzy–Gewicht. Diese Verbindungen heißen* **gekoppelt** *und ihre Gewichte heißen* **gemeinsame Gewichte**. *Eine analoge Bedingung gilt für Verbindungen, die zu derselben Ausgabeeinheit y_j führen.*

(v) *Sei $L_{x,R}$ die Bezeichung der Verbindung zwischen einer Eingabeeinheit x und einer Regeleinheit R. Für alle Regeleinheiten R, R' gilt $(\forall x \;\; L_{x,R} = L_{x,R'}) \Longrightarrow R = R'$.*

Diese Definition ermöglicht es, ein NEFPROX–System als ein gewöhnliches Fuzzy–System zu interpretieren. Jede innere Einheit repräsentiert eine Fuzzy–Regel. Die Bedingung (iv) erzwingt *gemeinsame* oder *verbundene* Gewichte. Würde diese Eigenschaft fehlen, könnten Fuzzy–Gewichte, die einen identischen linguistischen Term repräsentieren, während des Trainings unterschiedlich modifiziert werden. Wäre dies zulässig, so könnte jede Regel ihre eigenen individuellen Zugehörigkeitsfunktionen für Prämisse und Konklusion entwickeln. Dadurch wäre eine geeignete Interpretation der Regelbasis nicht möglich, was auf keinen Fall erwünscht ist. Die Bedingung (v) fordert, daß es keine Regeln mit identischen Prämissen im System gibt.

Diese Bedingungen sind sehr ähnlich zu den Definitionen von NEFCON bzw. NEFCLASS. NEFCON besitzt jedoch nur eine einzige Ausgabeeinheit, und NEFCLASS verwendet keine Fuzzy–Mengen für die Gewichtung der Verbindungen zu den Ausgabeeinheiten.

3 Das Lernverfahren

Für ein auf Funktionsapproximation beruhendes Problem läßt sich überwachtes Lernen einsetzen, da die Ausgabe für jede Eingabe aus der Menge der Beispieldaten bekannt ist (feste Lernaufgabe). Falls ein Fuzzy–System zur Approximation einer Funktion eingesetzt werden soll, kann Vorwissen genutzt werden. Das bedeutet, wenn für bestimmet Bereiche der Funktion bereits geeignete Regeln bekannt sind, so kann das Neuro–Fuzzy–System damit initialisiert werden. Die fehlenden Regeln müssen durch ein Lernverfahren gefunden werden. Falls kein Vorwissen existiert, beginnt ein NEFPROX–System mit leerer innerer Schicht und lernt alle Regeln inkrementell hinzu.

Der Lernalgorithmus für NEFPROX ist in der folgenden Definition beschrieben. Wir verwenden dreieckförmige Zugehörigkeitsfunktionen, die durch drei Parameter beschrieben werden:

$$\mu : \mathbb{R} \to [0,1], \quad \mu(x) = \begin{cases} \frac{x-a}{b-a} & \text{falls } x \in [a,b), \\ \frac{c-x}{c-b} & \text{falls } x \in [b,c], \\ 0 & \text{sonst.} \end{cases}$$

Die am weitesten rechts bzw. links gelegebenen Zugehörigkeitsfunktionen können zudem geschultert sein. Wir verwenden dreiecksförmige Fuzzy–Mengen aus Gründen der Einfachheit. Das Lernverfahren ist jedoch problemlos auf andere Formen von Zugehörigkeitsfunktionen erweiterbar. Als Defuzzyifikationsstrategie wird Center of Gravity (COG) oder Mean of Maximum (MOM) in den Ausgabeknoten verwendet.

Um den Lernprozeß zu starten, sind für alle Eingabegrößen beliebige Fuzzy–Mengen vorzugeben. Für die Ausgabegrößen ist dies nicht notwendig, da in diesem Fall Fuzzy–Mengen während des Lernvorgangs erzeugt werden können. Es ist jedoch möglich, wenn das Verfahren mit bestimmten Fuzzy–Mengen beginnen soll. Falls keine Fuzzy–Mengen für die Ausgabevariablen vordefiniert werden, ist eine anfängliche Breite für eine neu zu erzeugende Zugehörigkeitsfunktion anzugeben (hier: $|a - c|$).

Definition 3 (NEFPROX Lernalgorithmus) *Wir betrachten ein NEFPROX–System mit n Eingabeeinheiten $x_1, \ldots, x_n$, k Regeleinheiten $R_1, \ldots, R_k$, und m Ausgabeeinheiten $y_1, \ldots, y_m$. Weiterhin ist eine Lernaufgabe $\tilde{\mathcal{L}} = \{(\mathbf{s}_1, \mathbf{t}_1), \ldots, (\mathbf{s}_r, \mathbf{t}_r)\}$ mit r Mustern gegeben, wobei jedes Muster aus einem Eingabemuster $\mathbf{s} \in \mathbb{R}^n$, und einem Zielmuster $\mathbf{t} \in \mathbb{R}^m$ (erwünschte Ausgabe) besteht. Der Lernalgorithmus zur Erzeugung der k Regeleinheiten des NEFPROX–Systems besteht aus den folgenden Schritten* **(Strukturlernalgorithmus)***:*

(i) Wähle das nächste Muster $(\mathbf{s}, \mathbf{t})$ aus $\tilde{\mathcal{L}}$

(ii) Finde für jede Eingabeeinheit $x_i \in U_1$ die Zugehörigkeitsfunktion $\mu_{j_i}^{(i)}$ mit

$$\mu_{j_i}^{(i)}(s_i) = \max_{j \in \{1,\ldots,p_i\}} \{\mu_j^{(i)}(s_i)\}$$

(iii) Falls keine Regeleinheit R mit

$$W(x_1, R) = \mu_{j_1}^{(1)}, \ldots, W(x_n, R) = \mu_{j_n}^{(n)}$$

existiert, dann erzeuge eine solche Einheit und verbinde sie mit allen Ausgabeeinheiten.

(iv) Finde für jede Verbindung von der neuen Regeleinheit zu den Ausgabeeinheiten ein geeignetes Fuzzy–Gewicht durch das folgede Verfahren:
Aus allen Zugehörigkeitsfunktionen, die zur Ausgabeeinheit y_i gehören wähle $\nu_{j_i}^{(i)}$, so daß

$$\nu_{j_i}^{(i)}(t_i) = \max_{j \in \{1,\ldots,q_i\}} \{\nu_j^{(i)}(t_i)\}, \text{ und } \nu_{j_i}^{(i)}(t_y) \geq 0.5.$$

gilt. Gibt es keine derartige Fuzzy–Menge, dann erzeuge $\nu^{(i)}_{neu}$ so daß $\nu^{(i)}_{neu}(t_i) = 1$ gilt, füge sie zu den y_i zugeordneten Zugehörigkeitsfunktionen hinzu, und setze $W(R, y_i) = \nu^{(i)}_{neu}$.

(v) Falls es noch nicht verarbeitete Muster in $\tilde{\mathcal{L}}$ gibt, dann fahre mit Schritt (i) fort, ansonsten beende die Regelerzeugung.

(vi) Evaluiere die Regelbasis abschließend: Ermittle für jede Regel einen Mittelwert der Ausgaben aller Muster, die einen Treffergrad größer Null zu der jeweiligen Regel aufweisen. Ändere gegebenenfalls die entsprechende Ausgabe–Fuzzy–Menge in der Konklusion der jeweiligen Regel, falls der Mittelwert einen höheren Zugehörigkeitsgrad zu einer anderen als der aktuelle Ausgabe–Fuzzy–Menge aufweist.

Der überwachte Lernalogorithmus zur Adaption der Fuzzy–Mengen eines NEFPROX–Systems läuft zyklisch durch die Lernaufgabe $\tilde{\mathcal{L}}$ und wiederholt die folgenden Schritte, bis ein gegebenes Abbruckkriterium erfüllt ist **(Parameterlernalgorithmus)***:*

(i) Wähle das nächste Muster $(\mathbf{s}, \mathbf{t})$ aus $\tilde{\mathcal{L}}$, propagiere es durch das NEFPROX–System und bestimme den Ausgabevektor.

(ii) Für jede Ausgabeeinheit $y_i \in U_3$: Bestimme den Deltawert $\delta_{y_i} = t_i - o_{y_i}$

(iii) Für jede Regeleinheit R mit $o_R > 0$:

(a) Bestimme für alle $y_i \in U_3$ die Deltawerte für die Parameter a, b, c der Fuzzy–Menge $W(R, y_i)$ unter Verwendung einer Lernrate $\sigma > 0$:

$$\begin{aligned}
\delta_{b_i} &= \sigma \cdot \delta_{y_i} \cdot (c - a) \cdot o_R \cdot (1 - W(R, y_i)(t_i)),\\
\delta_{a_i} &= \sigma \cdot (c - a) \cdot o_R + \delta_{b_i},\\
\delta_{c_i} &= -\sigma \cdot (c - a) \cdot o_R + \delta_{b_i},
\end{aligned}$$

und wende die Änderungen auf $W(R, y_i)$ an, falls dies nicht gegen eine gegebene Menge Φ von Einschränkungen verstößt. (Beachte: Das Gewicht $W(R, y_i)$ kann mit anderen Verbindungen geteilt und in diesem Fall mehr als einmal verändert werden.)

(b) Bestimme den Deltawert

$$\delta_R = o_R(1 - o_R) \cdot \sum_{y \in U_3} (2W(R, y)(t_i) - 1) \cdot |\delta_y|.$$

(c) Bestimme für alle $x \in U_1$ die Deltawerte für die Parameter a, b, c der entsprechenden Fuzzy–Menge $W(x, R)$, falls $W(x, R)(o_x) > 0$ gilt:

$$\begin{aligned}
\delta_b &= \sigma \cdot \delta_R \cdot (c - a) \cdot (1 - W(x, R)(o_x))) \cdot \operatorname{sgn}(o_x - b),\\
\delta_a &= -\sigma \cdot \delta_R \cdot (c - a) \cdot (1 - W(x, R)(o_x))) + \delta_b,\\
\delta_c &= \sigma \cdot \delta_R \cdot (c - a) \cdot (1 - W(x, R)(o_x))) + \delta_b.
\end{aligned}$$

Dabei ist $\sigma > 0$ eine Lernrate. Wende diese Änderungen auf $W(x, R)$ an, sofern dies nicht gegen gegebene Einschränkungen Φ verstößt. (Beachte: Das Gewicht $W(x, R)$ kann mit anderen Verbindungen geteilt und in diesem Fall mehr als einmal verändert werden.)

(iv) Prüfe am Ende einer Epoche, ob das Abbruchkriterium erfüllt ist und halte in diesem Fall an. Andernfalls fahre mit Schritt (i) fort.

Der Strukturlernalgorithmus wählt Fuzzy–Regeln auf der Grundlage eines vordefinierten „Gitters“ aus, das durch die vorgegebenen Fuzzy–Mengen im Eingaberaum aufgespannt wird. Falls der Algorithmus mehr Regeln als gewünscht erzeugt hat, ist es möglich durch Auswertung individueller Regelfehler die schlechtesten Regeln wieder zu löschen.

In diesem Fall ist jedoch mit einer Verschlechterung der Funktionsapproximation zu rechnen. Jede gebildete Fuzzy–Regel faßt eine Menge von Stützstellen zusammen und repräsentiert sie als eine unscharfe Stützstelle. Wenn Regeln gelöscht werden, bedeutet dies, das einige Stützstellen der Lernaufgabe nicht mehr berücksichtigt werden. Falls der Parameterlernalgorithmus dies nicht ausgleichen kann, muß sich die Approximationsleistung verschlechtern.

Das Lernverfahren für die Zugehörigkeitsfunktionen entspricht einer einfachen Heuristik. Es verschiebt die Fuzzy–Mengen und verändert ihre Träger. Es ist leicht Einschränkungen Φ für den Lernvorgang zu spezifizieren. Zum Beispiel sollten sich benachbarte Fuzzy–Mengen nicht überholen. Für das Abbruchkriterium verwendet man am besten eine Validierungsmenge und beobachtet den Fehlerverlauf über dieser Menge. Wenn der Fehler nicht weiter absinkt, wird das Training beendet.

Zur Beschleunigung des Lernverfahrens kann eine adaptive Lernrate eingesetzt werden. Solange der Fehler abnimmt, wird die Lernrate erhöht. Steigt der Fehler wieder, wird sie abgesenkt.

4 Zusammenfassung und Ergebnisse

In diesem Beitrag wurde ein Lernverfahren für ein Neuro–Fuzzy–System vorgestellt. Das Verfahren ist in der Lage, die Struktur und die Parameter eines Fuzzy–Systems zur Approximation von Funktionen zu erlernen. Das resultierende NEFPROX–Modell ist eine Erweiterung der NEFCON– und NEFCLASS–Ansätze. Die Lernfähigkeiten des Modells werden zur Zeit noch eingehend untersucht. Die ersten Ergebnisse sind jedoch sehr ermutigend.

Als Beispiel für das Lernverhalten betrachten wir eine chaotische Zeitreihe, die durch die bekannte „Mackey–Glass Differentialgleichung“ gegeben ist:

$$\dot{x}(t) = \frac{0.2x(t-\tau)}{1 + x^{10}(t-\tau)} - 0.1x(t).$$

Das Ziel besteht darin, aus den Werten $x(t-18), x(t-12), x(t-6)$ und $x(t)$ den Wert $x(t+6)$ vorherzusagen. Die Daten wurden mit Hilfe eines Runge–Kutta–Verfahrens mit Schrittweite 0.1 und den Anfangsbedingunen $x(0) = 1.2$ und $\tau = 17$ erzeugt. Es wurden die 1000 Werte der Zeitpunkte $t = 118$ bis 1117 verwendet, wobei die ersten 500 Werte zum Training und die zweiten 500 Werte zur Validierung dienten.

Zur Approximation wurde ein NEFPROX–System mit vier Eingangs– und einer Ausgangsgröße erzeugt, wobei für jede Variable sieben Fuzzy–Mengen zugelassen wurden. Das Lernverfahren erzeugte 129 Regeln, und das NEFPROX–System hatte insgesamt 105 veränderbare Parameter. Als Defuzzifizierungsverfahren wurde die Mittelpunktmethode (MOM) gewählt, da diese Methode weniger Berechnungsaufwand als die Schwerpunktmethode (COG) verursacht, und die Wahl der Methode keinen wesentliche Einfluß auf die Approximationsleistung hatte. Als t–Norm wurde min und als t-Conorm max verwendet. Das resultierende NEFPROX–System entspricht somit einem gewöhnlichen Mamdani–Fuzzy–System.

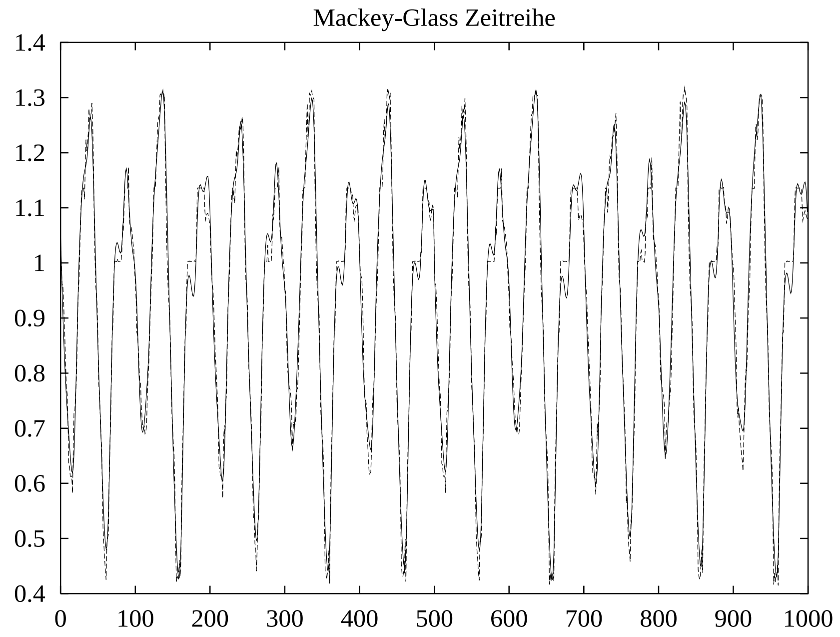

Abbildung 1: Approximation der Mackey–Glass Zeitreihe mit NEFPROX (216 Lernepochen)

Die Abbildung 1 zeigt die Approximationsleistung nach 216 Epochen (durchgezogene Linie = Originaldaten, gestrichelte Linie = NEFPROX). Die Werte 1 – 500 entsprechen den Trainingsmustern und die Werte 501 – 1000 der Validierungsmenge. Die Wurzel des mittleren quadratischen Fehlers (RMSE = root mean square error) ergibt für die Trainingsmenge 0.0315 und für die Validierungsmenge 0.0332. Die Trainingsdauer auf einem Linux–PC (80486-DX2 66 MHz) beträgt acht Minuten, auf einer SUN UltraSparc 75 Sekunden. Mit einem ANFIS–Modell mit zwei Fuzzy–Mengen pro Variable, 16 Regeln und insgesamt 104 veränderbaren Parameter, das ein Sugeno–Fuzzy–System implementiert, läßt sich zwar

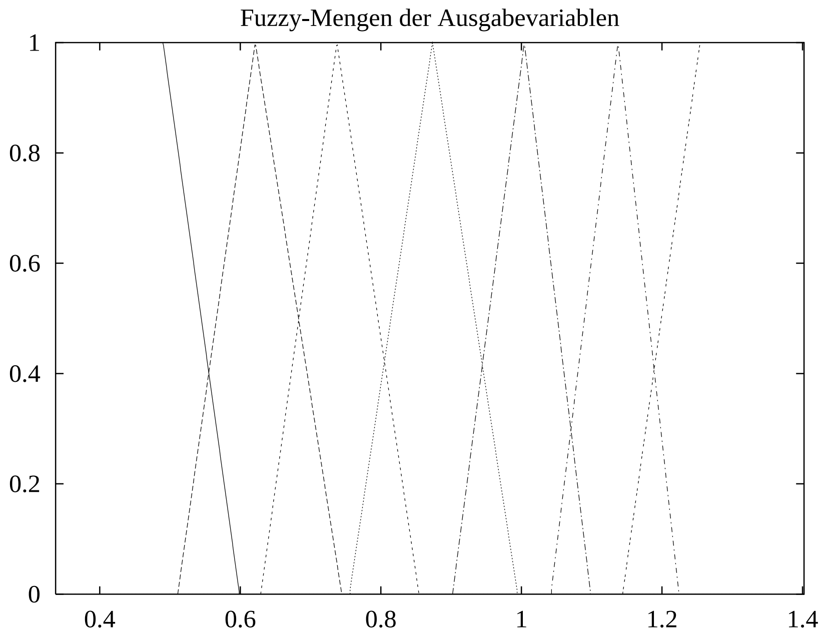

Abbildung 2: Fuzzy–Partition über der Ausgabevariablen nach dem Training

eine bessere Approximation erzielen (RMSE von 0.0016 bzw. 0.0015)[3] jedoch benötigt der Trainingsvorgang etwa die 15–fache Zeit auf derselben Rechnerarchitektur (18 Minuten auf einer SUN UltraSparc unter Verwendung der ANFIS–Software, die von Jang unter ftp.cs.cmu.edu in user/ai/areas/fuzzy/systems/anfis zur Verfügung gestellt wird).

Die weiteren Arbeiten am NEFPROX–Modell konzentrieren sich auf der Entwicklung besserer Strukturlernverfahren, um die Anzahl der Fuzzy–Regeln zu senken und so eine bessere Interpretation des Ergebnisses zu ermöglichen. Wünschenswert wäre auch eine vermehrte Regelerzeugung in Bereichen, in denen die Ausgaben stark variieren, um eine bessere Approximation zu erzielen. Die vorliegende Modellversion hat noch Probleme, Extremwerte sauber zu approximieren. Dieser Effekt läßt sich mit der Eigenschaft der Defuzzifizierungsverfahren erklären, die eher dazu neigen, Ausgaben in der Mitte als an den Rändern des Wertebereiches zu erzeugen.

Literatur

[1] J. J. Buckley. Sugeno type controllers are universal controllers. *Fuzzy Sets and Systems*, 53:299–303, 1993.

[2] Simon Haykin. *Neural Networks. A Comprehensive Foundation*. Macmillan College Publishing Company, New York, 1994.

[3] J.-S. Roger Jang. ANFIS: Adaptive-network-based fuzzy inference systems. *IEEE Trans. Systems, Man & Cybernetics*, 23:665–685, 1993.

[4] Bart Kosko. Fuzzy systems as universal approximators. In *Proc. IEEE Int. Conf. on Fuzzy Systems 1992*, pages 1153–1162, San Diego, March 1992.

[5] Rudolf Kruse, Jörg Gebhardt, and Frank Klawonn. *Fuzzy–Systeme, 2. erweiterte Auflage*. Teubner, Stuttgart, 1995.

[6] Detlef Nauck. Building neural fuzzy controllers with NEFCON–I. In Rudolf Kruse, Jörg Gebhardt, and Rainer Palm, editors, *Fuzzy Systems in Computer Science*, pages 141–151. Vieweg, Braunschweig, 1994.

[7] Detlef Nauck, Frank Klawonn, and Rudolf Kruse. *Neuronale Netze und Fuzzy–Systeme, 2. erweiterte Auflage*. Vieweg, Wiesbaden, 1996.

[8] Detlef Nauck and Rudolf Kruse. NEFCLASS – a neuro–fuzzy approach for the classification of data. In K.M. George, Janice H. Carrol, Ed Deaton, Dave Oppenheim, and Jim Hightower, editors, *Applied Computing 1995. Proc. of the 1995 ACM Symposium on Applied Computing, Nashville, Feb. 26–28*, pages 461–465. ACM Press, New York, February 1995.

[9] Detlef Nauck and Rudolf Kruse. Designing neuro–fuzzy systems through backpropagation. In Witold Pedrycz, editor, *Fuzzy Modelling: Paradigms and Practice*, pages 203–228. Kluwer, Boston, 1996.