【翻訳】海外におけるNN+FSシステムの研究と応用

Nauck, Detlef
Technical University of Braunschweig

Kruse, Rudolf
Technical University of Braunschweig

髙木, 英行[訳]
九州芸術工科大学音響設計学科

出版情報：1996-04-01. 朝倉書店
バージョン：
権利関係：

第 8 章

海外におけるＮＮ＋ＦＳシステムの研究と応用

Detlef Nauck　Rudolf Kruse
Technical University of Braunschweig
Department of Computer Science
Bueltenweg 74 – 75, D–38106 Braunschweig, Germany
Tel: + 49.531.391.3155, Fax: +49.531.391.5936, Email: nauck@ibr.cs.tu-bs.de
（翻訳：高木英行）
（815 福岡市南区塩原４丁目９－１）
（九州芸術工科大学 音響設計学科）
（Tel 092-553-4555,　FAX 092-553-4569　takagi@kyushu-id.ac.jp）

本章ではニューラルネットとファジィシステムの融合化技術（ＮＮ＋ＦＳシステム）の研究と日本国外の応用事例を概説する。先見的知識が使え、例えばファジィコントローラなどに使える形で言語ルールが取り出せる適応システムを構築するために、これまでニューラルネットとファジィシステムを組み合わせる方法がいくつか提案されている。これらのうち本章では２つのタイプを取り上げる。１つはニューラルネットとファジィシステムの個々の個性を活かして組み合わせる協調モデル、もう１つは２つの技術コンセプトを使った新しいアーキテクチャを生み出すハイブリッドモデルである。

キーワード 海外動向、ニューロ的ファジィモデル、ハイブリッドモデル、応用、ソフトウェア開発環境、モデル選択指針

8.1　はじめに

ニューラルネットとファジィシステムを組合せるという考えがここ数年来さまざまな形で試されてきた。ファジィコントローラが提案された当初から、設計したコントローラが与えられたタスクで最もうまく働くように適応的に変化させる技術の必要性が認識されていた。しかし今日まで、はじめから完全な記述ができないファジィシステムの性能を最大にするような理論的基盤は確立されていなかった。これまではチューニングという名の経験的な手法で性能を最大にするようパラメータ変更してきただけである。チューニングは特に対象が複雑な場合には手間のかかる方法である。本章の読者は多少なりともファジィシステムについてご存じのものとして論を進めることとする。入門編が必要な方は文献 30, 31) をご参照いただきたい。

当初は、プロセスの知識に基づいて自身のパラメータを更新し自分自身を調整するファジィコントローラが考えられた。適応型とか自己組織型ファジィコントローラと呼ばれるこの方法は例えば文献 45, 50, 46) に見られる。この分野の解説としては文献 9) がある。この種の適応モデルは通常知識ベースの方法を用い、ニューラルネットは用いない。

ニューラルネット、別名コネクショニストシステム、は人間の脳のある一面をモデル化するために作られたものである。ニューラルネットは、重み結合に基づいて信号を変換する単純な処理単位（ニューロン）からなる。ある種のニューラルネットは汎用の近似モデルであり、いかなる連続関数でもあらゆる要求精度で近似可能である。また、この近似はサンプルからの学習で行われる。ニューラルネットの入門編としては文献 1, 37, 58) を参照されたい。

ニューラルネットの最大の特長であるデータからの学習能力は、しかしながら、ニューラルネット内部がブラックボックス的であるがために、魅力が多少落ちる。一般的にはネットワークを初期化するための先見的知識も使えず、かつ、学習後のネットワークからルールの形で知識を取り出すこともできない。一方、ファジィシステムは知識、すなわち言語的ルールを扱うことができるが学習能力はない。一方の短所はまた一方の長所でもある。ここからニューラルネットとファジィシステムを組合せ、長所を失うことなく他方の短所をカバーしようとする考えが自然に生まれて来る。

これらのモデルはニューロ的ファジィシステムとかニューロ・ファジィとか呼ばれ、この研究領域は “ファジィシステムの仲間内 ”では日増しに重要性と関心が高まっている。ほとんどのニューロ的ファジィシステムは制御関連に偏っている。ファジィ制御の分野では重要な流れとして捉えられているが、しかし、この分野に限定されることなく他の分野、例えばデータ解析の分野などでも使われている。また、ニューラルネットの学習の最適化に関するモデルもある [34)]。これらのモデルは、「ファジィニューラルネット」とか「ファジィ – ニューロシステム」とでも名付けられるべきものである。多層パーセプトロンのためのバックプロパゲーション学習則の学習率をファジィ制御的手法を用いて動的に決定する方法は文献 19) に述べられている。ここでは、これらの方法についてはこれ以上深く述べない。ここでは、ニューラルネットワークでファジィシステムパラメータを決定することに主眼を置いて述べる。

8.2 ニューロ的ファジィモデル

ニューロ的ファジィモデルは、通常次の４つのタイプに分類される。

(a) ファジィ集合を学習するタイプ：ニューラルネットはサンプルデータからファジィ集合を学習するために用いられる。具体的には、メンバーシップ関数の形状を規定するパラメータを学習したり、ニューラルネットでメンバーシップ関数を近似しファジィシステムに組み込まれたりする。学習は通常オフライン、すなわちシステムが実際のタスクに適用される前に行われ、最急降下法が使われる。ファジィルールはこれとは別に作成しなければならない。

(b) ファジィルールを学習するタイプ： ニューラルネットはサンプルデータからファジィルールを学習する。通常オフラインのクラスタリングアルゴリズムで行われ、ニューラルネットは自己組織特徴写像や同類のモデルと winner–takes–all 学習則や適応ベク

トル量子化を組み合わせて使われる。ファジィ集合は学習前に定義しておかなければならない。ニューラルネットの代わりにファジィクラスタリングも使えるかもしれない[6)]。

(c) ファジィ集合を適応的に変化させるタイプ： ニューラルネットは既に設計されたファジィシステムのメンバーシップ関数の形状パラメータを変更することに使われる。性能を測るエラー指標があればオンライン、すなわちシステムが実際に稼働中に適用することも可能である。時にはニューラルネットが使われていないように見える場合もあるが、この場合はコネクショニスト学習アルゴリズムが直接ファジィシステムに応用されている。

(d) ファジィルールのスケーリングタイプ： ニューラルネットはファジィルールの重み付けを決定するために使われる。そのためにはファジィ集合とファジィルールとが事前に決められていなければならない。学習はオンラインでもオフラインでも行われる。重み付けは通常ルールの「重要度[54, 28)]」として表される。しかしこれには意味論的な問題が内在している（第8.6節参照）。ファジィルール出力をスケーリングすることは後件部メンバーシップ関数を変更することと等価である。

ニューロ的ファジィモデルに続いて、ニューラルネットをファジィシステムの前処理部または後処理部として用いるモデルがある（第１章の図○○）。これらのニューロ－ファジィ組み合わせモデルはシステムの入力変数が直接計測できず多くの値の組み合わせで表さなければならない場合には有用である。このように、ニューラルネットを適応「情報圧縮器」として機能させることができる。一方、ファジィシステムの出力が直接プロセスに使えず、他のパラメータと組み合わせて使わなければならない場合もある。このような場合にもニューラルネットが使える。

ここで使われるニューラルネットはブラックボックスなので、ルールを取り出すという意味ではニューロファジィ組合わせモデルには難がある。しかし、学習結果からルールを取り出す必要がなく、先見的知識が使える簡単な適応システムが必要な場合には有用な方法である。

本章の主旨からは、これらの方法はニューロ的ファジィモデルには分類しないでおく。なぜならニューラルネットはファジィシステムのパラメータ決定には使われていないからだ。したがってこれ以上の議論は差し控えるが、後の応用の節でもう一度取り上げる。

タイプ（ａ）のモデルには例えば野村らの研究がある。彼らのモデルは教師あり学習を用い、予め用意されたファジィルールベースに基づいた菅野らのファジィモデルのファジィ集合を決定する[42)]。その後この方法の改良版が文献5) と37) で述べられている。ルール前件部にはパラメータ化された三角形メンバーシップ関数が用いられている。

$$\mu_r^{(i)}(\xi_i) = \begin{cases} 1 - \dfrac{2\,|\xi_i - a_r^{(i)}|}{b_r^{(i)}} & a_r^{(i)} - \frac{b_r^{(i)}}{2} \le \xi_i \le a_r^{(i)} + \frac{b_r^{(i)}}{2} \text{の場合} \\ 0 & \text{その他} \end{cases}$$

このメンバーシップ関数はファジィルール R_r で使われる変数 $\xi_i \in X_i$ のファジィ集合を定義している。システムの出力は $\eta = \sum_{r=1}^{k} \tau_r \; y_r \; / \; \sum_{r=1}^{k} \tau_r$ で表される。ここで τ_r はルール R_r を

どの程度満たしているかを表し、$\tau_r = \prod_{i=1}^{n} \mu_r^{(i)}(\xi_i)$ で与えられる。また、実数値 y_r はルール R_r の出力である。

野村らの方法では $r \neq r'$ に対して $\mu_r^{(i)} \neq \mu_{r'}^{(i)}$ であるようなファジィ集合を持つことを許す。すなわち、同じ言語変数であってもルールが違えば異なる意味を持ち得るわけである。バックプロパゲーション [48] を用いてメンバーシップ関数のパラメータを決定するため、t–ノルム演算を通して得られる関数が微分可能であるような t– ノルムを使って前件部の評価をする必要がある。このため、ルールをどの程度満たしているかを求めるために MIN ではなく代数積が使われている。

この方法の問題点は、同じ言語変数に対して別のルールでは違う表現を学習しようとすることによって生じる意味論的な問題である。この問題を解決する方法は、メンバーシップ関数 $\mu_1^{(i)}, \ldots, \mu_{p_i}^{(i)}$ $(i = 1, \ldots, n)$ を用いて X_i を普通に分割することである。つまり、変数 $\xi_i \in X_i$ の各言語値がファジィ集合の形で１つだけの表現を持つようにするのである。$Ant(R)$ をルール R の前件部のファジィ集合とし、η_p を実際の出力、η_p^* を望ましい出力としよう。野村らのオリジナルの方法の代わりに、次の方法を用いてメンバーシップ関数パラメータを更新すればよい（$\sigma_a, \sigma_b, \sigma_y$ は学習率）。

$$\Delta_p a_{j_i}^{(i)} = \left(\sum_{r:\mu_{j_i}^{(i)} \in Ant(R_r)} \tau_r \left(y_r - \eta_p \right) \right) \frac{\sigma_a}{\sum_{j=1}^{k} \tau_j} \left(\eta_p^* - \eta_p \right) \frac{2 \operatorname{sgn}(\xi_i - a_{j_i}^{(i)})}{b_{j_i}^{(i)} \ \mu_{j_i}^{(i)}(\xi_{i,p})},$$

$$\Delta_p b_{j_i}^{(i)} = \left(\sum_{r:\mu_{j_i}^{(i)} \in Ant(R_r)} \tau_r \left(y_r - \eta_p \right) \right) \frac{\sigma_b}{\sum_{j=1}^{k} \tau_j} \left(\eta_p^* - \eta_p \right) \frac{1 - \mu_{j_i}^{(i)}(\xi_{i,p})}{b_{j_i}^{(i)} \ \mu_{j_i}^{(i)}(\xi_{i,p})}.$$

変数 y_r の修正量 $\Delta_p y_r = \sigma_y \tau_r \cdot (\eta_p^* - \eta_p) \ / \ \sum_{j=1}^{k} \tau_j$ は変化させてはいけない。

この方法を用いれば、学習後のファジィシステムには同じ言語値を示す異なるファジィ集合は存在しない。彼らの論文では三角メンバーシップ関数を用いているため微分不可能な点をどう扱うかという問題もあるが、この問題は微分不可能な点では値を更新しないことで解決できる。

Pedrycz と Card によって試された Kohonen の自己組織化特徴写像 [27] の言語的表現は、コネクショニスト学習則によってファジィルールを作り出すことが可能 [44] で、これはタイプ（ｂ）のニューロ的ファジィモデルに該当する。

制御対象には、特徴写像が n 個の入力ノード $v_1, \ldots, v_n$ からなるような n 個の（入出力）変数があり、出力層は $n_1 \times n_2$ の２次元写像からなるものとしよう。競合学習にはプロセス状態のベクトルからなるサンプル集合 $\mathcal{L}$ と正しい制御出力値が必要である。これらの値と結果として得られる重みはすべて $[0, 1]$ である。特徴写像の自己組織化の後は、変数 $\xi_i \in X_i$ を表す入力ノード v_i に対応する重み行列 $\mathbf{W}_i$ からなる１つの２次元写像で各プロセス変数が記述される。

p_i 個のメンバーシップ関数 $\mu_{j_i}^i$ が各変数 $\xi_i \in X_i$ 用に定義された後、これらのメンバーシップ関数は写像変換に使われる。これは、各変数に対して１つのファジィ集合を選ぶことで行われる。この選択は言語記述 $\mathcal{B}$ として示されている。この変換された写像は次に、行列 $\mathbf{D}^{(\mathcal{B})} = [d_{i_1,i_2}^{(\mathcal{B})}]$ を求めるために共通部分が求められる。この行列は、学習結果と以下の序列 $(j_1, \ldots, j_n)$ を表す言語記述 $\mathcal{B}$ とに互換性があることを示す：

$$\mathbf{D}^{(\mathcal{B})} = \bigcap_{i:i\in\{1,\ldots,n\}} \mu_{j_i}^{(i)}(\mathbf{W}_i), \quad d_{i_1,i_2}^{(\mathcal{B})} = \min_{i:i\in\{1,\ldots,n\}} (\mu_{j_i}^{(i)}(w_{i_1,i_2,i}))$$

行列 $\mathbf{D}^{(\mathcal{B})}$ はファジィ関係、$d_{i_1,i_2}^{(\mathcal{B})}$ はノード v_{i_1,i_2} における言語記述 $\mathcal{B}$ への寄与度を表す。$\mathbf{D}^{(\mathcal{B})}$ の最大のメンバーシップ値は $\mathcal{B}$ と学習結果との互換性の度合、と解釈される。行列 $\mathbf{D}^{(\mathcal{B})}$ をその α– カット $\mathbf{D}_\alpha^{(\mathcal{B})}$ で表すと、メンバーシップ値が少なくとも α である出力ノードのサブセットが得られ、各 v_{i_1,i_2} に対して、$||\mathbf{w}_{i_1,i_2} - \mathbf{x}_{k_0}|| = \min_{\mathbf{x}:\mathbf{x}\in\mathcal{L}} ||\mathbf{w}_{i_1,i_2} - \mathbf{x}||$ であるようなパターン $x_{k_0} \in \mathcal{L}$ を探すことで、各 $\mathbf{D}_\alpha^{(\mathcal{B})}$ はパターンサブセット $X_\alpha^{(\mathcal{B})} \subseteq \mathcal{L}$ を包含し、明らかに $\alpha_1 \geq \alpha_2$ の時，$X_{\alpha_1}^{(\mathcal{B})} \subseteq X_{\alpha_2}^{(\mathcal{B})}$ である。

もし十分大きな α_0 があれば、集合 $X_{\alpha_0}^{(\mathcal{B})}$ は言語記述 $\mathcal{B}$ で記述されるクラスの代表点の集合と見ることができる。もし行列 $\mathbf{D}^{(\mathcal{B})}$ の α– カット $\mathbf{D}_\alpha^{(\mathcal{B})}$ が空でないならば、各言語記述 $\mathcal{B}$ は１つのクラスタを有効に記述していると言える。各 $\mathcal{B}$ は１つの言語ルールを表し、言語変数の各組み合わせで前件部・後件部を記述することで完全なファジィルールベースが構築できる。

この方法はまたどのパターンがクラスタ、すなわちルールに属さないかも示す。どこにも属さないパターンが非常に多い場合は、メンバーシップ関数の選択が不十分であるかもしれない。この方法の問題点は、α_0 と出力層ユニット数の決定にある。学習アルゴリズムは Kohonen の特徴写像に基づいており、学習率を小さくして収束させる。しかし、学習結果がパターン集合の構造を正しく示しているという保証はない。もう一つ、学習結果はパターンを伝播させた順序に依存する。

この方法の長所はパターン空間の構造が考慮されていることと、メンバーシップ関数から得られる情報が学習結果に最もマッチするルールの決定に完全に用いられる点である。

ファジィルールベースを作成する他の方法は Kosko によって示された FAM (Fuzzy Associative Memory) モデルに基づく方法である [28]。有限集合 $X = \{x_1, \ldots, x_m\}$ を考えよう。ファジィ集合 $\mu : X \to [0,1]$ は m– 次元の超立方体 $I^m = [0,1]^m$ の上の１点として見ることができる。するとファジィルール

$$R\text{: } \textbf{If } \xi_1 \textbf{ is } A_{j,1}^{(1)} \wedge \ldots \wedge \xi_n \textbf{ is } A_{j,n}^{(n)} \textbf{ Then } \eta \textbf{ is } B_j$$

は写像 $R : I^{m_1} \times \ldots \times I^{m_n} \to I^s$ と解釈することができる。FAM は前件部に１つの μ と後件部に１つの ν からなるファジィルール (μ, ν) を格納することに使われる。ここで μ と ν はメンバーシップ関数であり、$\mu_i = \mu(x_i)$ かつ $\nu_i = \nu(y_i)$ としよう。すると、FAM は結合行列 $\mathbf{W} = [w_{i,j}] = \mu \circ \nu,\ \min(\mu_i, \nu_j)$ で定義され、これはファジィ関係 $\rho : X \times Y \to [0,1]$ を表す。$\mathbf{W}$ はファジィ *Hebb* 行列であり、その計算は相関 MIN 符号化（*correlation minimum encoding*）と呼ばれる [28]。その想起は $\nu = \mu \circ \mathbf{W},\ \nu_j = \max_{i:i\in\{1,\ldots,m\}} \min(\mu_i, w_{i,j}) = \min(v_j, \text{height}(\mu))$ で定義される。したがって、$\text{height}(\mu) \geq \text{height}(\nu)$ が満たされるならば、想起結果は常に正しい。これは、通常のファジィ集合、すなわち $(\exists\, x \in X)\ \ \mu(x) = 1$ が使われるならば、想起誤りがないことを意味する。

FAM は１つのファジィルールしか格納できない。同時に複数のルールを格納すれば想起結果に多くの誤りが生じることになるからである。内部の演算のために、パターン間の干渉は通常のニューラル連想記憶に比べ厳しいものになる。複数入力変数があって前件部の中に結合があるルールを扱うには複数の FAM を用い、各想起を MIN で結合する。

FAM をパラレルに並べてファジィコントローラを構成することができる。このような FAM システムへの入力は２値ベクトルで、各ベクトルには値が１ (fuzzy singletons) である要素が１つだけある。出力は $[0,1]^s$の ベクトルとして表現されるファジィ集合である。非ファジィ化を行う要素があれば、出力もまた２値ベクトルになる（BIOFAM: Binary Input–Output FAM）。 FAM に組み込まれたルールはさらに $[0,1]$ の重み付けがされる。したがって、 FAM はタイプ（ｄ）のモデルである。

ルールの重みの決定はファジィルールベースを作る間に行われる。学習は微分競合学習（DCL: *differential competitive learning*）と呼ばれる一種の適応ベクトル量子化で行われる [28)]。このために、 n 個の入力ユニットと $k = p_1 \cdot \ldots \cdot p_n$ 個の出力ユニットを持つニューラルネットが使われる。入力ユニットは全出力ユニットに結合され、出力ユニットは側抑制のトポロジーを実現するため相互に結合されている。これは変数上での分割を定義しなければならないことを意味する。すなわちファジィ集合が分かっていないといけない。メンバーシップ関数の形状は学習結果に影響を及ぼさない。メンバーシップ値が０でない領域のみが考慮されるからである。各出力ニューロン、すなわち重みベクトルはファジィルールを表す。

学習にはプロセス状態のサンプルデータと正しい制御動作が必要である。ある１つの出力ユニットと全入力ユニットを学習した後、どのファジィ集合が当該入力ユニットとその出力ユニット間の重み係数用に正のメンバーシップ値を示すかがチェックされる。この方法では、重みベクトルは１つまたは複数のルールに使うことができる。ファジィルールの重みは重みベクトルの数、すなわち相対頻度によって決まる。 FAM システムが作られシステムの演算が行われている間、ルールの重み係数の更新やルールの追加削除によって学習が続けられる。適応 FAM システムはタイプ（ｂ）と（ｄ）を組み合わせる方法である。

この方法の最も大きな問題はルールの重み付けである。まず初めに、この方法には FAM システムの解釈における意味論的問題がある（第 8.6節を参照のこと）。もし学習データが、例えばプロセスをよく検討して制御する有能な操作者を観察して得られたものならば、ぎりぎりの状態や極端な状態は多分データにうまく反映されていないであろう。もし操作中に学習が行われるならば、これらの極端な状態の重みは徐々に小さくなり、長時間の実行後はこのような状態が制御できないシステムになってしまう。

Kosko は極端な状態を処理するルールをマニュアルで入れることを勧めているが、これにはプロセスの知識が必要であるし、またそのような知識が得られるとは限らない。ユーザーはまた得られたルールベースには矛盾がない、すなわち同じ前件部で異なる結論を導くようなルールは存在しないことにも留意しなければならない。

FAM 行列は学習されることがなく、ファジィルールの格納という観点からすると何かしら効率的ではない。ＮＮ＋ＦＳシステムがこのアプローチに適するのはルール重みの適応性と学習能力があるからである。しかし学習アルゴリズムは FAM 表現に依存してはいるわけではなくどのようなファジィシステムにも使える。学習アルゴリズムも、 Pedrycz と Card のモデルで使われたように、データ集合の構造に組み込まれたトポロジーを考慮しているわけではない。事前に定義されたファジィ集合に組み込まれた情報も完全には使われていない。

一方 FAM システムは単純で、いくつかの適応性の特徴を示すモデルを実現することが容易である。このため、この方法はいくつかの市販のファジィシェルに使われている。

8.3 ハイブリッドＮＮ＋ＦＳモデル

ハイブリッドＮＮ＋ＦＳモデルはニューラルネットとファジィシステムの概念を用いることで新しいアーキテクチャを作り出す。モデルは通常ファジィシステムとして解釈可能であり、また特別の励起関数 (activation function) と伝播関数を持ったニューラルネットとも見ることができる。

Berenji の ARIC (Approximate Reasoning based Intelligent Control) モデル [3] は特別なニューラルネットを複数用いるハイブリッドＮＮ＋ＦＳモデルである。 ARIC アーキテクチャは適応批判（adaptive critics）の概念と特別なニューロコントローラの補強学習 (reinforcement learning)[55] を用い、 Barto らのニューラルモデル [2] をファジィ制御の分野で汎用化したものである。 ARIC は ASN (Action Selection Network) と AEN (Action state Evaluation Network) の２つのニューラルモデルから構成されている。

ASN 自体は２つの３層フィードフォワードネットワークから構成されている。そのうちの１つはファジィコントローラを直接構成したものであり、もう１つは制御ネットワークの出力値を変更するために使う確信値（confidence value）を求めるのに用いる。制御ネットワークの入力層はプロセスの状態変数を表す。各ユニットは該当変数の三角メンバーシップ関数を内蔵し、ファジィルールに相当する中間層ユニットに結合される。ルールの前件部は該当ルールのノードへの重み結合を伝播する言語値で表現される。すなわち、入力ユニットはどのメンバーシップ値がどの結合に伝播されるかを知っておかなければいけない。メンバーシップ値は結合重みでスケーリングされ、 MIN 演算でルールノード内で結合される。

後件部のメンバーシップ値はルールノード内に内蔵されている。 ARIC は関数が単調である塚本のファジィ集合 [32] を用いている。正値の各メンバーシップ値に対して領域の要素は１つしか存在しないないため、この方法では各ルールはクリスプ値を出力する。ルール出力は、中間層と出力層の間の重み結合後、加算し出力ユニットへ伝播される。この出力値は次に、確率的制御量修正部（*stochastic action modification*）と呼ばれるものでスケーリングされる。これは、制御ネットワークと同じ重みにさらに入出力間の結合重みを持った ASN の２番目（非ファジィ）のニューラルネットの値に基づいている。

AEN ネットワークはプロセス状態を予測するよう学習する適応批判エレメントである。プロセス制御が失敗したかどうかを AEN に知らされる外部補強信号に基づいて、ネットワークは ARIC システム全体の重み係数を適応的に修正することに用いる内部補強信号を計算する。大きな内部補強信号がある場合（良いプロセス状態）は、重み係数は出力値への貢献が大きくなるよう更新される（報償:rewarding）。プロセス制御が失敗した時は、貢献が少なくなるよう重み係数を更新する（罰則:punishment）。内部補強が単に小さい場合は、確率的制御量修正部を大きくして、システムがより良い出力値をランダムに生成するようにする。この方法は出力値に白色雑音を加えることに似ており、文献 2) で行われている。

ARIC ネットワークには中間層があるため、補強学習だけで学習を行うことができない。 ARIC は内部補強を最適化するように働き、そのためにバックプロパゲーションの考えを学習に組み込んである。

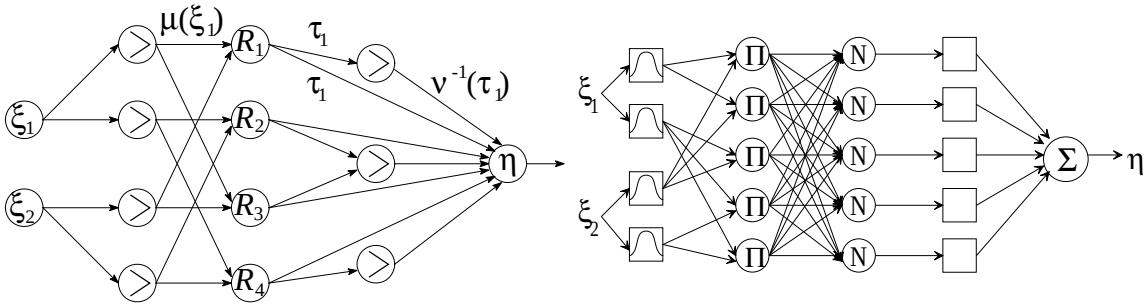


図 8.1: ２つのハイブリッド NN ＋ FS モデル： GARIC の ASN [4]（左）と ANFIS の構造 [23]（右）

ARIC の短所はその複雑なニューラルネット構造にあり、制御ネットワークの重みを更新することによって意味論的問題が起こることである。前件部の重みを変えるということはメンバーシップ関数をスケーリングすることと等価であり、通常のファジィ集合ではなくなる。重み係数が $[0, 1]$ である保証はないので、もはやメンバーシップ関数とは見なせない関数になるやもしれない。後件部の重み係数を変更することはメンバーシップ関数をシフトすることと等価なので、ファジィ集合の非零領域に望ましくない変更を生じさせるかもしれない（例えば、 “Positive Big” が負の値にドリフトするかもしれない）。また同じ言語値が別のルールでは違って表現される可能性がある [38, 36]。

Berenji と Khedkar は ARIC のこの問題を解決すべく ASN を修正し、 Generalized ARIC を意味する GARIC モデルを提案した（図 8.1左）。今度の ASN は、前件部と後件部のメンバーシップ関数を持つユニットの層を２つ加えたニューラルネットワーク１つからのみ構成されている。ルールベースは結合として表され、適応的に修正される重み係数はない。学習では三角メンバーシップ関数のパラメータを修正更新していく [4]。

AEN の学習アルゴリズムは同じだが、 ASN は勾配法で内部補強信号が最適化される。このためには、ルール前件部を評価するための微分可能な関数が必要になる。つまり、ここでは MIN 演算は使われない。その代わりに GARIC では *soft MIN* という $t-$ノルムではない演算子を使う。 ARIC と同様、学習アルゴリズムには各ルールからのクリスプ出力値が必要である。この場合、通常の MAX などでファジィ集合を統合（aggregate）してから非ファジィ化するという処理をしなくてもよい。 GARIC では各ルールからクリスプ値を得るためにローカル最大平均法（LMOM: *local mean of MAX*）と呼ばれる処理を用意している。この処理ではメンバーシップ関数が非対称の場合、常に通常の最大平均法（MOM）とは違う結果が得られる。

学習アルゴリズムは内部補強信号を最適化するため勾配法を用いる。しかし、この信号と GARIC からの制御出力との依存性ははっきりとは分からないので、学習は経験的ないくつかの仮定を用いなければならない。さらに、経験的に解決しなければならない問題は、メンバーシップ関数に微分不可能な３つの点があることである。

学習方法は内部補強信号の変化によって異なり、信号が一定になったら学習は終了する。学習が最適になった時にこの状態になる。しかし、プロセスが一定に保たれるが必ずしも最適ではない状態の場合にも起こり得るかもしれない。したがって GARIC の学習とは失敗しないために学習するのであって、最適な状態になるために学習するわけではない。つまり最適ではなく局所最適になるかも知れない。この種の問題は文献 43) で述べられている。

ARIC と GARIC は共にメンバーシップ関数のみ学習可能で、システムのルールベースは別の手段で決めておく必要がある。このモデルではメンバーシップ関数を初期化しておくことも必要で、その数を変えることはできない。ハイブリッドＮＮ＋ＦＳモデルには通常この制約がある。これらのアプローチの特長は状態での制御値が分かっている必要がないことである。モデルは試行錯誤を通じて学習していく。このことは、プロセスシミュレーションが可能だとか、オンライン学習が可能だとか、さらにはプロセスで失敗しても危険ではない、などの条件が暗に求められる。

この種の他のアプローチとして Jang の ANFIS(Adaptive Network based Fuzzy Inference System) モデルがある [22, 24]。 ANIFS は各層毎に異なる励起関数を持つ特別な５層 feedforward ニューラルネットワーク（入力は層に数えられていない）で菅野らのファジィモデルを構成する（図 8.1右）。 ANFIS は

$$R_r\text{: } \textbf{If } \xi_1 \textbf{ is } A_{j_1}^{(1)} \wedge \ldots \wedge \xi_n \textbf{ is } A_{j_n}^{(n)} \textbf{ Then } \eta = \alpha_0^{(r)} + \alpha_1^{(r)}\xi_1 + \ldots + \alpha_n^{(r)}\xi_n$$

の形のルールを組み込んでいる。

学習アルゴリズムでメンバーシップ関数のパラメータと後件部を α_j を調整する。ネットワークの結合は重みを持たず、修正可能なパラメータは４層目のノードで表される。 ANFIS は勾配法ではなく α_j 値を求める数式に基づいた教師あり学習を行う。 ANFIS は汎用ニューラルネット構造を使った簡単な方法なので、 *Stuttgart Neural Networks Simulator* SNNS 3.0 [7, 57] で行われたように、汎用ニューラルネットシミュレータで容易に実現できる。この点は応用の観点から ANFIS を魅力あるものにしている。

GARIC の ASN 部に似た構造が文献 52) でも用いられている。 FUN(FUzzy Net) と呼ばれるこのモデルは結合とメンバーシップ関数のパラメータをランダムに変更する確率的学習法を用いてメンバーシップ関数とファジィルールを学習している。コネクショニスト学習法ではなく、確率的探索手法で学習が行われるため、ＮＮ＋ＦＳモデルの分類としては中心には位置するものではない。

NEFCON(NEural Fuzzy CONtroller) モデルは補強学習アルゴリズムを用い、メンバーシップ関数とファジィルールを学習する [40, 41]。これは最新のＮＮ＋ＦＳモデルの１つであり、したがって次節で詳しく述べる。

8.4 NEFCON モデル

NEFCON はファジィパーセプトロンに基づく neural fuzzy controllers である [35]。このシステムは３層のユニットからなり、層間の結合はファジィ集合で重み付けられている [39, 40]。NEFCON 用の学習アルゴリズムはファジィルールのようにメンバーシップ関数を学習できる [39, 40]。

ファジィ誤差を用いることで、上述した ARIC や GARIC のような他の方法で必要とされる適応批判エレメントを用いることなく補強タイプの学習アルゴリズムを定義することができる。ルールベースの学習はルールの削除で行われる。これはオンライン学習が可能であり、前述したようなクラスタリング手法に使うサンプルデータは不要であることを意味する。

NEFCON モデルは入力層、（中間）「ルール」層と出力層とからなる汎用の３層ファジィモデル [35] から導き出される。層間の結合はファジィ集合で重み付けられる。各層はいくつ

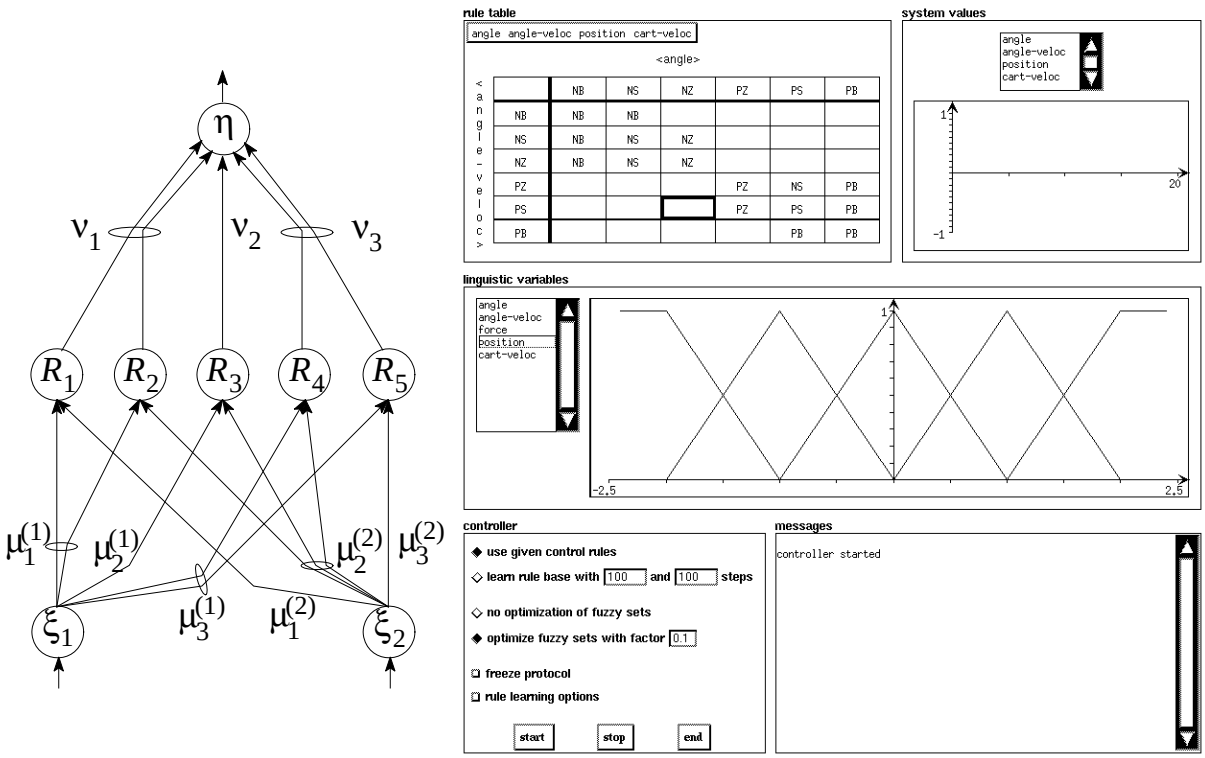


図 8.2: ２入力の NEFCON システムと NEFCON–I ソフトウェアのユーザーインタフェース

かのユニットを持ち、中間「ルールユニット」は励起関数として t– ノルムを用いる。出力ユニットはファジィ集合を組み合わせ非ファジィ化を行う。入力ユニットは単に入力値を入れるだけでそれ以上の処理は行わない。

入力値がネットワークを伝播することと、通常のファジィシステムで使われている IF-THEN ファジィルールで推論を行うこととは等価である。ファジィパーセプトロンの学習は、理想的出力が分かっている場合はその教師データと実際の出力との差に基づいて行われ、理想的出力が分かっていない場合は、制御状態と誤差の関係を記述したファジィルールから得られたファジィ誤差に基づいて行われる。この誤差はメンバーシップ関数を修正するようネットワーク構造を逆伝播する。

NEFCON システム (図 8.2参照) は特殊な３層ファジィパーセプトロンで、次の仕様を持つ：

(i) 入力ユニットは $\xi_1, \ldots, \xi_n$ で記述され、中間ルールユニットは $R_1, \ldots, R_k$ と記述示され、１つの出力ユニットは η と記述される。

(ii) ξ_i と R_r の各結合は $A_{j_r}^{(i)}$ $(j_r \in \{1, \ldots, p_i\})$ と言語ラベル付けされる。

(iii) ユニット R_r と出力ユニット η の各結合は B_{j_r} $(j_r \in \{1, \ldots, q\})$ と言語ラベル付けられる。

(iv) 同じ入力ユニット ξ_i に同じラベルで結合されている場合は、常に同じファジィ重みを持つ。これらの結合は共通結合と呼ばれる。出力ユニット η につながる結合にも同様な条件が当てはまる。

(v) $L_{\xi,R}$ を入力ユニット ξ とルールユニット R との結合のラベルとする。全ルールユニットに対して、R, R' $(\forall \xi\ L_{\xi,R} = L_{\xi,R'}) \Longrightarrow R = R'$ が成り立つ。

以上の定義によって NEFCON システムは、各中間ユニットがファジィ IF-THEN ルールを表すファジィシステムであると解釈できる。条件 (iv) は共通重み *(shared weights)* が必須であることを規定している。もしこの条件が欠落すれば、同じ言語で表されるファジィ重みが学習中に異なって展開されることも有り得る。そうなれば、NEFCON システムはファジィルールベースとして受け入れられなくなってしまう。条件 (v) は同じ前件部を持つルールが存在しないためのものである。この条件に固執しないネットワークを強 - 制約 *NEFCON* システムと呼ぶ。このシステムは可能なルールをすべて取り込んでおり、条件 (v) で不要なルールユニットを削除することで言語ルールベースを学習する [40, 41]。

次に、入力層と中間層の間の結合メンバーシップ関数 $\mu_j^i \quad (i \in \{1,\ldots,n\}, j \in \{1,\ldots,p_i\})$ を前件部とし、中間層と出力層の間の結合メンバーシップ関数 $\nu_j\,(j \in \{1,\ldots,q\})$ を後件部としよう。前件部は 3 つのパラメータ a, b, c を持つ三角メンバーシップ関数で後件部にはパラメータ d, e を持つ塚本の単調メンバーシップ関数 [32] を用いる。これらは次のように定義される。

$$\mu_j^{(i)}(x) = \begin{cases} \dfrac{x - a_j^{(i)}}{b_j^{(i)} - a_j^{(i)}} & x \in [a_j^{(i)}, b_j^{(i)}] \text{の場合,} \\ \dfrac{c_j^{(i)} - x}{c_j^{(i)} - b_j^{(i)}} & x \in [b_j^{(i)}, c_j^{(i)}] \text{の場合,} \\ 0 & \text{その他,} \end{cases}$$

$$\nu_j(y) = \begin{cases} \dfrac{d_j - y}{d_j - e_j} & (y \in [d_j, e_j] \wedge d_j \le e_j) \\ & \vee (y \in [e_j, d_j] \wedge d_j > e_j) \text{ の場合,} \\ 0 & \text{その他,} \end{cases}$$

ここで、$a_j^{(i)}, b_j^{(i)}, c_j^{(i)} \in \mathbb{R}$、$a_j^{(i)} \le b_j^{(i)} \le c_j^{(i)}$、$d_j, e_j \in \mathbb{R}$ である。

NEFCON システムは 1 つの制御変数 η を入力し n 個の状態変数 $\xi_1, \ldots, \xi_n$ を出力する動的システム S を制御する。NEFCON の性能は、

IF ξ_1 がほぼゼロかつ ξ_2 がほぼゼロ， **THEN** 誤差は少ない。

のようにファジィルールで定義したファジィ誤差で評価される。ここで ξ_1 と ξ_2 は動的システムの状態変数であり、それぞれ NEFCON に入力される。誤差がファジィルールで定義されているので、その値も制御出力 η と同様に定義できる。すなわち、2 つ目の NEFCON システムを同じタスクに使うことができる。ファジィルールから得られる定義された誤差は学習アルゴリズムで使われる。

定義 **1** n 個の入力ユニット $\xi_1, \ldots, \xi_n$、k 個のルールユニット $R_1, \ldots, R_k$、出力ユニット η からなる *NEFCON* システムにおいて、メンバーシップ関数を自動設計するファジィ誤差バックプロパゲーションを以下のように定義し、終了条件に達するまで繰り返す。

(i) *NEFCON* 出力 $o_\eta = (\sum_R o_R \cdot t_R) \,/\, (\sum_R t_R)$ を求める。ここで $o_R = min\{\mu_R^{(1)}(\xi_1), \ldots, \mu_R^{(n)}(\xi_n)\}$ である。また $t_R = \nu_R^{-1}(o_R)$ はルールユニット R のクリスプ出力値で、単調関数 ν_R

の逆関数として直接計算できる。$\mu_R^{(i)}$ はルールユニット R を入力ユニット ξ_i との結合、ν_R は R と出力ユニット η への結合である。

(ii) 出力値を S に入力し S の新しい状態を決定する。

(iii) S の状態からファジィ誤差 E を決定する。E はファジィ誤差を記述するファジィルールから求められる。

(iv) S の新しい状態に対する理想的出力値 $\eta_{\rm opt}$ の符号を求める（理想的出力値そのものはもちろんわからないが、符号ならばわかるはず）。

(v) その出力値 η に含まれる各ルールユニット出力 R_r の出力 t_r を各々求め、各 R_r $(r \in \{1,\ldots,k\})$ のファジィルール誤差を求める*:*

$$E_{R_r} \stackrel{def}{=} \begin{cases} -o_{R_r} \cdot E & \mathrm{sgn}(t_r) = \mathrm{sgn}(\eta_{\rm opt})\text{の場合} \\ o_{R_r} \cdot E & \text{その他} \end{cases}$$

(vi) 拘束条件 $\Psi(\nu_{j_r})$（後述）の下で、パラメータ修正量 $\Delta\, d_{j_r} \stackrel{def}{=} \sigma \cdot E_{R_r} \cdot |d_{j_r} - e_{j_r}|$ を求め、メンバーシップ関数 ν_{j_r} $(j_r \in \{1,\ldots,q\},\ r \in \{1,\ldots,k\})$ を修正する。ただし学習率 σ は正数。

(vii) 拘束条件 $\Psi(\mu_{j_r}^{(i)})$（後述）の下で、パラメータ修正量 $\Delta\, a_{j_r}^{(i)} \stackrel{def}{=} -\sigma \cdot E_{R_r} \cdot (b_{j_r}^{(i)} - a_{j_r}^{(i)})$ と $\Delta\, c_{j_r}^{(i)} \stackrel{def}{=} \sigma \cdot E_{R_r} \cdot (c_{j_r}^{(i)} - b_{j_r}^{(i)})$ を求め、メンバーシップ関数 $\mu_{j_r}^{(i)}$ $(i \in \{1,\ldots,n\},\ j_r \in \{1,\ldots,p_i\},\ r \in \{1,\ldots,k\})$ を修正する。

以上の繰り返しでファジィ誤差 E が一定値以下になったならば、これをもって学習終了とすることができるかもしれない。メンバーシップ関数の拘束条件 Ψ は、例えば常に $a \le b \le c$ であるとか、ファジィ集合間には一定の重なりがあるとか、等である。

この学習アルゴリズムはメンバーシップ関数の修正にのみ用いられ、ファジィルールベースが分かっている場合に適用可能である。一部のルールしか分からない場合やまったくルールが分からない場合にはファジィルールを学習するよう学習アルゴリズムを拡張することも可能である。この場合、はじめに強 - 制約 NEFCON システムで学習を始め、学習過程で最も大きな累積誤差を出したルールユニットを削除するようにする。分かっているルールがないときには、システムはファジィ分割の結果として得られる可能な全ルール $N = q \cdot \prod_{i=1}^{n} p_i$ 個のルールノードから学習を始める（各変数毎のファジィ集合の数は事前に決定しておかなければならない）。次の定義において、$\mathcal{R}$ は全ルールユニットの集合を、$Ant(R)$ はルールユニットの前件部を、$Con(R)$ はルールユニットの後件部を示す。

定義 **2** $(\forall R, R' \in \mathcal{R})\ (Ant(R) = Ant(R') \wedge Con(R) = Con(R')) \implies R = R'$ である $N = q \cdot \prod_{i=1}^{n} p_i$ 個のルールユニットと、p_i 個のファジィ集合に分割される変数を表す n 個の入力ユニット $\xi_1,\ldots,\xi_n$ と、q 個のファジィ集合に分割される変数を表す 1 つの出力ユニット η からなる強 - 制約 *NEFCON* システムにおいて、不要なルールユニットを削除する拡張ファジィ・バックプロパゲーション・アルゴリズムは次の手順で行われる。

(i) 各ルールユニット R_r 毎に、カウンタ C_r を 0 に初期化しておく $(r \in \{1,\ldots,N\})$。
一定の反復回数 m_1 に対して、次のステップを実行する。

(a) 現在の状態 S における *NEFCON* 出力 o_η を求める。

(b) 各ルールユニット R_r 毎に、全体出力 o_η を構成する各ユニット出力 t_r を求める $(r \in \{1, \ldots, N\})$。

(c) 現在の入力値に対して符号 $\mathrm{sgn}(\eta_{\mathrm{opt}})$ を求める。

(d) $\mathrm{sgn}(t_r) \neq \mathrm{sgn}(\eta_{\mathrm{opt}})$ のルールユニットを削除し、 N の値を更新する。

(e) $o_{R_r} > 0$ である R_r のカウンタ C_r に 1 を加える。

(f) 出力 o_η を状態 S に適用し、新しい入力値を求める。

(ii) 各 R_r 毎にカウンタ Z_r を 0 に初期化する。
一定の反復回数 m_2 に対して、次のステップを実行する。

(a) すべてのサブ集合 $\mathcal{R}_j = \{R_r | Ant(R_r) = Ant(R_s),$
$(r \neq s) \wedge (r, s \in \{1, \ldots, N\})\} \subseteq \mathcal{R}$ から任意に 1 つのルールユニット R_{r_j} を選ぶ。

(b) 選んだルールユニットと状態 S だけで *NEFCON* 出力 o_η を求める。

(c) o_η を S に適用し新しい入力値を求める。

(d) 総合出力 $(r_j \in \{1, \ldots, N\})$ に対する選ばれた各ルールユニット R_{r_j} の t_{r_j} を求める。

(e) 新しい入力値を用いて $\mathrm{sgn}(\eta_{\mathrm{opt}})$ を求める。

(f) 選ばれた各ルールの誤差 $E_{R_{r_j}}$ をそれぞれのカウンタ Z_{r_j} に加える。

(g) $o_{R_{r_j}} > 0$ である選ばれた各ルールユニット R_{r_j} に 対し、 C_{r_j} に 1 を加える。

$R_{r_j} \in \mathcal{R}_j$ に対して $Z_{r_j} < Z_{s_j}$ ならばルールユニット R_{s_j} をネットワークから削除する。ここで、 $\mathcal{R}_j$ はサブ集合である。更に、 $C_r < \frac{m_1 + m_2}{\beta}$, $\beta \geq 1$ であるルールユニット R_r を削除し、 N の値を更新する。

(iii) 残った $k = N$ 個のルールユニットを持つ *NEFCON* システムにファジィ誤差バックプロパゲーションを適用する（定義 **1**参照）。

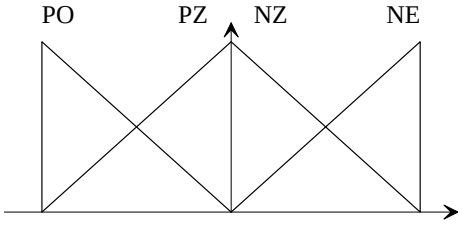
このルール学習アルゴリズムの考え方は、今あるルールを試してみて評価することにある。このテストをパスしないルールユニットはネットワークから削除される。最初の段階では、理想的出力と符号が異なるルールユニットはすべて削除される。第 2 段階では、アルゴリズムは同じ前件部を持つ各ルールセットから 1 つづつルールを選び残りのルールを削除する。この手順により、強 - 制約 NEFCON システムから通常の NEFCON システムが得られる。第 3 段階ではメンバーシップ関数が最適化される。

多くの変数で多くのファジィ集合が定義されている場合は、ルール学習アルゴリズムの計算コストは非常に高くついてしまう。したがって、すべての可能なルール組み合わせから始めなくてもよいような部分的な知識をできるだけ導入すべきである。可能なすべてのルールから順次削除する方法は、ある入力状態についてまったく知識がない場合だけにすべきである。このように、初期のルールユニット数は削減される。

8.5 NEFCON–I – ＮＮ＋ＦＳソフトウェア

対話型グラフィックシミュレーション環境 NEFCON–I （I は InterViews を意味する）はX ウィンドウ上で動作し、パブリックドメインのグラフィック・インターフェース・ツール InterViews を使って作られている。 NEFCON–I は SUN ワークステーション上に移植してあり、インターネットを通じてバイナリィコードもソースコードも無料提供されている[1]。

ユーザーはメンバーシップ関数とルールをファイルからも読みだせるし、グラフィク・エディタで定義することもできる。 NEFCON–I は動的システムのシミュレーション部を含んだプログラムとやりとりする。シミュレーションとしては例えば、 Runge–Kutta 法を使い数式と定数が文献 2) に示されている倒立振子などのプログラムを使う。



$\dot{\theta}$ \ θ	NE	NZ	PZ	PO
NE	NE	NE	NE	PZ
NZ	NE	NZ	PZ	PO
PZ	NE	NZ	PZ	PO
PO	NZ	PO	PO	PO

図 8.3: ファジィ誤差を定義するための θ と $\dot{\theta}$ のファジィ集合、および、ファジィ誤差ルール表

角速度 θ̇ ＼ 角度 θ

	nb	nm	ns	nz	pz	ps	pm	pb
nb	nb	nb						
nm	nb	nm	nm	ns				
ns	nb	nm	ns	nz				
nz		ns	ns	nz				
pz					pz	ps	ps	
ps					pz	ps	pm	pb
pm					ps	pm	pm	pb
pb							pb	pb

角速度 θ̇ ＼ 角度 θ

	nb	nm	ns	nz	pz	ps	pm	pb
nb								
nm			nm	nm				
ns		ns	nm	nz				
nz		nb	nb	nz				
pz					pz	pb	pb	
ps					pz	pb	pz	
pm					pm	pz		
pb								

図 8.4: 学習前（左）と学習後（右）の NEFCON のルールベース

シミュレーション結果

本節では倒立振子に NEFCON を適用してその学習能力を示そう。制御曲面上の変化を示せるよう、角度 $\theta \in [-90, 90]$ と角速度 $\dot{\theta} \in [-200, 200]$ についてのみ考える。以下に示す全実験のファジィ誤差は図 8.3に示すファジィ集合とルールで定義した。

実験は図 8.4左に示すルールベースから始めた。図 8.5では θ のメンバーシップ関数の学習前と後を示した。初期分割ではコントローラは倒立振子を安定に倒立させることができな

[1] NEFCON–I は、 anonymous ftp で `ftp.cs.ibr.tu-bs.de` (134.169.34.15) に接続し、ディレクトリィ `pub/local/nefcon` から入手可能である。 また World Wide Web (WWW) 上の `http://www.cs.tu-bs.de/ibr/projects/` でも入手可能である。 1995 年 7 月現在、上記アドレスは有効である。

かった。何回かの失敗とランダムな角度で再試行した後、最終的なファジィ集合が作られ、振子を安定に倒立させることができるようになった。SUN 2 SPARCstation で約２分、2,000 回の繰り返し後、この安定状態にたどりついた。

図 8.6は学習前と後の制御曲面を示している。学習後の制御曲面は最初の時に比べ大変滑らかになっており、良い制御ができることがわかる。

２番目の実験はルールベースの学習についてである。これには第１回目の実験で学習したメンバーシップ関数を使う。6,000 回の学習後、図 8.4右に示すルールベースが得られた。コントローラはこのルールベースで倒立振子を立てることができ、しかもその後メンバーシップ関数を数 100 回新しい状況に適応させた結果、±20 度傾けても戻して安定にすることができた。

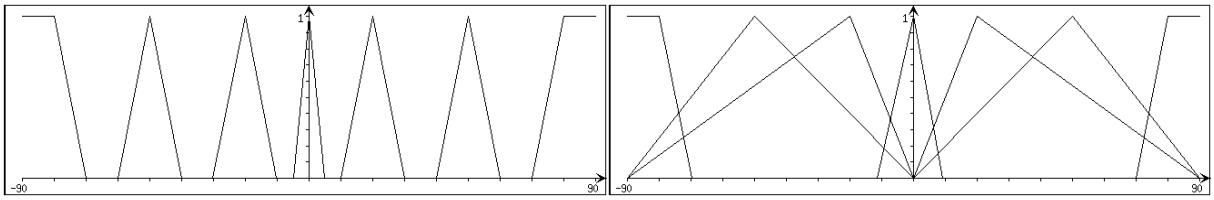


図 8.5: 学習前（左）と学習後（右）の θ のメンバーシップ関数

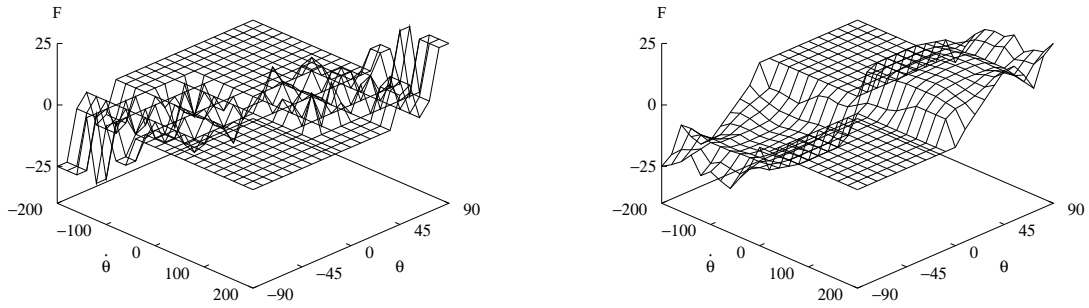


図 8.6: 学習前（左）と学習後（右）の制御曲面

その他のソフトウェア

NEFCON–I は筆者が保守管理しているので詳細に説明したが、もちろんこれだけが入手可能なＮＮ＋ＦＳソフトウェアという訳ではない。インターネットを通じて入手可能なプログラムとして他に *FuNeGen 1.0* がある。これはデータからファジィルールを獲得しバックプロパゲーション学習則でメンバーシップ関数をチューニングするものである。MS-DOS マシン用のこのソフトは Saman Halgamuge が保守管理しており、限定試用版が入手可能である[1] FuNeGen 1.0 はＮＮ＋ＦＳシステム FuNe I の考えに基づいている [20, 21]。

[1] anonymous FTP で `obelix.microelectronic.e-technik.th-darmstadt.de` の `/pub/neurofuzzy` から入手可能である。拡張版が同じ ftp サイトから提供される予定であり、文献 16, 18, 17) に基づいた各種のファジィシステムの設計手法が選択できるようになる。したがって Read.Me ファイルを最初に読まれることをお勧めする。

この他にＮＮ＋ＦＳソフトウェア製品もある。ツール *NeuFuzz* は National Semiconductor から入手可能である [26]。 NeuFuzz はファジィコントローラを構築するもので、バックプロパゲーションで学習する多層 feedforward ニューラルネットで構成されている。学習によって菅野らのファジィモデルを構成するパラメータが得られる。このモデルは、各ルールを生成しながら正規分布型のメンバーシップ関数とファジィルールを学習する。

その他のソフトウェア製品としてデータ解析に使うもので純粋なファジィモデルとニューラルネット以外にＮＮ＋ＦＳモデルを加えたものがある。このソフトは *Data Engine* といい、ドイツ Aachen の MIT GmbH から入手可能である。

8.6 意味論的側面

ＮＮ＋ＦＳモデルの応用を考えるならば、モデルの意味論的側面を知っておくべきであろう。本章で述べたいくつかのモデルは、学習で適応を始めたらファジィシステムに焼き直せないような方法でパラメータを更新する。

ルールの重みはしばしばルールの「重要度」と解釈されることがあるが、実際にはこのような見方は妥当ではない。重要度が低いルールがあった場合、そのルールは何か希な場合のみ使われるとか、使われなくても大したことはないと思われがちで、ルールの結論部がある特定の範囲にのみ考慮されるべきであるとは考えられないことがある。しかしこの点は、前件部のファジィ集合を定義する際に折込み済みなのだ。ファジィルールを重み付けするということと結論部を変更するということはまったく等価であるのも関わらず、しばしばファジィモデルの領域での話が置き忘れられている。異なるルールの重みには、ルールベースの中で違うように表現するため、異なる言語値が与えられる。これは普通望ましいことではない。

書き直しが重要でなければこのような問題は避けられる。先見的知識が扱える適応システムが必要な場合のみ、ＮＮ＋ＦＳモデルは有用である。この場合ならニューラルネットのみのシステムであっても構わない。しかしファジィルールの形に書き直し可能な構造があれば、マニュアルによる変更も可能であろうし、安全性の観点からも、または標準のファジィハードウェアが使うこともできるという点からも興味のあるところである。書き直し可能な構造はまた、知識獲得ツールとして用いることができる利点がある。

シミュレーションまたは実際のプロセスでオンライン学習が必要な可能な場合、ハイブリッドＮＮ＋ＦＳモデルが使える。これらのモデルを使うと、今まで制御できなかったプロセスを制御できるコントローラを設計できるかもしれない。しかも部分的な先験的知識も使える。これらのモデルは以後いつまででも一定の適応性を持ち続けるわけで、この点からも興味深い。 NEFCON のようなアプローチはファジィルールとメンバーシップ関数の両方を学習することができ、意味論的側面にも注意を払っている。

もしプロセス制御のたくさんのデータが使えるならば、ファジィルールが分からない時のファジィ集合の決定に、またはファジィ集合が分からない時のファジィルールの決定に、このデータをニューロ的ファジィモデルと一緒に使うことができる。この場合、ファジィクラスタリングのような他の手法も検討の対象になる。

8.7 応用

本節では最近のＮＮ－ＦＳ応用を紹介する。適当な関連の文献がない場合は、詳細な情報が得られるよう電子メールまたはWWWのアドレスを記載する。

本章執筆時点では、日本以外でのニューラルネットまたはファジィ技術の応用はようやく始まったばかりであり、それがためにＮＮ－ＦＳアプローチは非常に少ない。多くのＮＮ－ＦＳ応用はいまだ試作レベルであり、それらの多くも前述した意味でニューロファジィ組合わせの類であると思われる。これはよく分かっている技術のみを用いた方が扱いやすいことによる。代表的な例として、バイオ製薬プロセスの品質制御のためのニューロファジィ組合わせが述べられている[13)]。このシステムはセンサ情報を扱う２つのニューラルネットを持ち、ファジィシステムがこの情報と他の情報とを統合している。最初のニューラルネットは自己想起ネットワークで圧力、流れ、温度のセンサデータを確認するために使われる。これはセンサ信号からノイズを取り除くことで行われる。２番目のニューラルネットは現在のプロセス状態を２つのクラス（「制御中」と「制御外」）に分けることに使われる。このネットワークはプロセス中のエラーが発生したことは検出できるが、状態の診断は完全にはできない。完全な診断は、ファジィ化されたニューラルネットの出力と付加情報を入力するファジィルールベースで行われる。

ファジィシステムの出力がニューラルネットの入力となる構成で、人間の顔を認識するニューロファジィ組合わせの応用が文献25)に述べられている。ここでは、横顔のシルエット画像の中の重要な点の位置検出のためにファジィ論理ベースの顔の特徴量検出アルゴリズムが述べられている。これらの点で、目、鼻の先端、鼻の下、唇、顎の位置が決まる。これらの点は１０次元の特徴ベクトルで、ART2ネットワーク[8)]への入力となる。このネットワークは特徴量に基づいて111のデータベース画像を識別できた。

オランダのTewnte大学の*Neuro Fuzzy Center*では、Enschede高校と共同でＮＮ－ＦＳシステムをベルトコンベア制御に応用した。ファジィコントローラはベルトコンベアの速度と張り具合を一定の範囲内に保つよう制御する。ニューラルネット学習則は最適性能になるようコントローラのメンバーシップ関数を調整するために使われる。この学習はオフラインで行われる。これはニューロ的ファジィモデル（ａ）の応用事例である[1]。

ドイツHamburgのゴミ焼却場の制御が文献29)で紹介されている。制御システムはファジィ手法を用いてパワー、火の位置、最適燃焼を制御する３つの部分からなる。設計したファジィ制御システムを調整して最適化するためにＮＮ－ＦＳ的手法が使われている。この場合、KoskoのFAMアプローチでファジィルールの尤度を決定している（ニューロ的ファジィタイプ（ｄ））。

文献56)には多指ロボットハンドの制御にニューロ的ファジィモデルが紹介されている。このシステムは“Karlsruhe Dextrous Hand”でデモンストレーションを行い、穴にくさびを入れるタスクを行った。このシステムはタスクの解釈と実行計画を行う知識ベース部と、握力の短期局所的な補正を行うファジィ制御部と、手にかける力の順応とファジィ制御部のファジィ推論ルールの学習に用いるニューラルネット部から成っている。ニューラルネット部は握りのパラメータを長期的な適応変化のために用いられる。

トラッキング問題の制御に使う自由度Nのロボットマニュピュレータ用コントローラの設

[1]詳細はEric Schol氏 (schol@cs.utwente.nl) にお問い合わせいただきたい。

計にファジィ的ニューラルネット (ハイブリッドＮＮ＋ＦＳモデル) が使われている [47]。このファジィ的ニューラルネットには特別の AND と OR ニューロンが中間層と出力層に使われている。これらの励起関数は t– ノルム $\top(a,b) = ab$ と t– コノルム $\perp(a,b) = a + b - ab$ で表される。学習は教師あり勾配法で行われる。ファジィ的ニューラルネットへの入力は誤差と誤差変化量からなり、出力はＰＤタイプのローカルレギュレータの調整に使われる。

文献 15) には、モータの回転良否判定に用いたハイブリッドＮＮ＋ＦＳ法が述べられている。ＮＮ＋ＦＳシステムは２つのモジュールから構成され、１つはメンバーシップ関数の学習に、もう１つはファジィルールの学習に使われる。各モジュールは２層のニューロンからなり、最初のモジュール出力が２つ目の入力になっている。システムはバックプロパゲーション法で学習される。学習後、ファジィ集合は最初のモジュールのサブネットワーク（各入力変数毎に１つのネットワーク）を評価することで得られる。２つ目のモジュールを評価することで、システムからファジィルールを抽出することが可能である。このシステムには先見的知識が必要でないが、適当なメンバーシップ関数を選び第１のモジュールのサブネットワークをうまく初期化すれば学習が楽になる。このＮＮ＋ＦＳシステムは電気モータのベアリング摩耗不良の検出に応用された。システムへの入力はモータ速度と電流で、出力はモータが「不良」「並」「良」のいずれかであるかを判別する。

スイスの Lausanne にある EPFL のコンピュータサイエンス学部の Laboratory of Microinformatics (LAMI) では、 ANFIS 風のＮＮ＋ＦＳモデル [14] を同研究所で開発された移動ロボット Khepera [33] のための障害物回避システムを作成することに応用した[1]。

ANFIS の他の応用事例が文献 49) に述べられている。ここでは「特別」な日、例えば祭日や休日の電力消費予測に ANFIS ベースのシステムを適用している。昨年の統計データをシステムの学習に用いた。精度良い負荷予測は電力システムのより良い資源利用計画の助けとなる。

医療診断支援システムもＮＮ＋ＦＳシステムの有望な応用と思われる。フィンランドのÅbo Akademi 大学では GeDeMeDeS （a Generic System for Developing Medical Decsicion Support）の研究を行っている [10]。入力データ（患者の測定値や症状）は前処理でファジィ化される。この処理には専門家の知識が反映され、その後のニューラルネットによるデータ処理の負荷軽減が可能になる。 Eklund はデータ解析支援と臨床薬学の診断モジュール開発支援ツール *Diagai D*を提案した [11]。このシステムでは１層のニューラルネットで医療データベースからのファジィ化した入力データを処理する。このニューラルネットは多層のネットワークに比べ学習が非常に容易である。１９９３年現在このツールはまだ開発途中である [12]。

日本がファジィ、ニューラルネット、そしてＮＮ＋ＦＳ技術を民生機器へすばやく応用したことはよく知られている。しかし、その他の国々でも追いつき始めた兆候が見られる。ドイツ企業の AEG はニューラルネットを使って得られた「尤度」で重み付けしたファジィルールで制御する洗濯機 (“Öko–Lavamat 6953”) – タイプ（ｂ）のニューロ的ファジィアプローチ – を開発した。ニューラルネットは色々な実験から得られたデータに基づいてファジィルールのための尤度を学習することに使われる。最終的なルールベースは１５９のルールからなり、３日間で設計できた [51]。この洗濯機は従来機より少ない水で洗濯が可能である。

[1] 詳細情報は、 WWW page の html://lamiwww.epfl.ch/khepera/index.html で入手されるか、 WWW page の html://diwww.epfl.ch/w3lami/Team_Lami/JelenaGodjevac.htmlor で Jelena Godjevac 氏に問い合わせられたい。

8.8 モデル選択のためのガイドライン

前章までに、いくつかのＮＮ＋ＦＳモデル、いかに動作するか、そしていくつかの応用について検討してきた。以上を踏まえて、与えられたタスクにふさわしいモデルをどのように選ぶべきかというガイドラインについて検討してみたい。これらのガイドラインはあくまで枠組みであって、実際の応用を選択する時には具体的な応用を常に考慮する必要がある。初めに、応用が予定される時に心に留めておかねばならないＮＮ＋ＦＳシステムの主な特徴をいくつかまとめてみよう。

ＮＮ＋ＦＳモデルの構造は言語的知識（ファジィルール）で規定できる。これは、ニューラルネットの中間層のユニット数をいくつにしたらよいか、のように、何かを推察する必要がないことを意味する。しかしながら、もし部分的な知識しかなく全体構造が決定できないとしても、ＮＮ＋ＦＳモデルは利用可能である。ＮＮ＋ＦＳモデルはさらに、構造（ルール）の学習能力のようなものを持っているからである（例えばNEFCONなど）。

言語的知識を用いると最適解近傍からＮＮ＋ＦＳシステムの学習を始めることができる。これは、通常のニューラルネットと比べると学習時間を相対的に短くすることができることを意味する。しかし学習データ数が少なくてもよいというわけではない。

ＮＮ＋ＦＳモデルの学習能力は、通常のファジィシステムでは必要な人手による調整からユーザーを解放する。しかし、学習が成功するとか理想的な状態にたどり着くことが常に保証されているわけではないことを、常に心に留めておくべきである。

多くのＮＮ＋ＦＳモデルの持つファジィシステムへの書き直し可能な構造によって、言語的ルールを抽出することが可能になる。しかし、この点はどの学習方法を用いたかに大きく依存する。すなわち、無制限なパラメータの更新が行われたとすると、システムが持っていた初めの意味論的な一面は失われてしまうかもしれない。

今度は、適当なＮＮ＋ＦＳモデルを異なる環境下で選ぶためのガイドラインについて述べよう。これらのガイドラインは本章でこれまで述べてきた考察をまとめたものである。

まず第１に述べたいことは、意味論的側面が考慮されるべきであるという点である。同じシステム内では同じ言語変数を異なるメンバーシップ関数で表わすべきではないし、ルールの重みは注意して使うべきである。なぜならこの重みの解釈が常にはっきりしているわけではないからである。通常これらは、普通のファジィ集合ではなくなり、同じ言語変数に対して暗に複数の表現を与えることになる。

既にあるファジィシステムを修正する場合は、メンバーシップ関数のみ修正すべきである。ルールベースの修正は実際、新しいファジィシステムを作ることになるからである。これまでルールベースを削除したり追加したりするＮＮ＋ＦＳシステムはなかった。しかし、新しいルールベースを作り出すモデルはある（後述参照）。メンバーシップ関数の正しい形を学習するためには、ニューロ的ファジィアプローチを使うべきである。もしくは、ファジィシステムをハイブリッドＮＮ＋ＦＳモデルに変換し、学習後元のファジィシステムに戻すことも可能。もちろん、学習データやオンライン学習機能が必要である。

既にあるファジィシステムを修正するが意味論的側面はあまり重要でない場合は、単純なルール重みの解決法が利用可能。この機能は簡単に実現できるので、既に多くの商用ファジィソフトウェアに組み込まれている。しかし、上述した欠点には甘んじなければならないであろう。また、ニューロファジィ組合せモデルを使うことも検討する。すなわち、ファ

ジィシステムの出力を修正するためにニューラルネットを使う。この方法ではファジィシステム自体のパラメータは修正されず、ニューラルネットはブラックボックスとして振舞う。つまり、システム全体は完全にファジィシステムには書き直せない。

一から新しいファジィシステムを作る場合は、メンバーシップ関数とファジィルールが学習でき、意味論的問題を回避できるハイブリッドＮＮ＋ＦＳモデルを利用する。そのモデルはまた部分的な知識をも扱えるべきである。このことによって、部分的なルールベースを定義し、足りないルールを学習し、学習データから何でも柔軟に学習することができるようになる（例えばNEFCON を参照のこと）。このようなハイブリッドモデルが使えない、または使うべきでない場合は、いくつかのニューロ的ファジィモデルを試してみることだ：この時は、初期メンバーシップ関数を推察し、学習データからファジィルールを学習するようニューロ的ファジィモデルを使い（ファジィクラスタリング [6] も検討に値する）、メンバーシップ関数のパラメータを更新するようニューロ的ファジィモデルを使う。

部分的な知識や学習データがあり、意味論的側面とファジィシステムへの書き直し可能かどうかを気にしなければ、ニューラルネット環境で容易に実現でき標準的な学習方法が使えるハイブリッドモデルを試してみる（例えば、高木，林のＮＮ駆動型ファジィ推論 [53] など）。

多くの変数を扱わなければならない場合は、構造化アプローチを試してみること。すなわち、適当なモデルに問題の一部を担当させ、それらを組み合わせて全体システムとする。また、（ＮＮ＋）ＦＳシステム用の新しい変数を作るためにニューラルネットを使ってみること。つまり、複数の入力を組み合わせてより複雑な１変数に変換してみる。例えば、部屋の雰囲気という１変数は温度、湿度、照明、などを入力するニューラルネットの出力として得られ得る。

今日まで多くのＮＮ＋ＦＳシステムが提案されてきた。しかし、日本の産業界を除いて、実応用の経験は非常に乏しく、どのように産業応用が図られたかは、通常公開されていない。

もしＮＮ＋ＦＳモデルの応用経験が少ないならば、意味論的にすっきりしていて自由度の少ない簡単な方法を選択した方がよい。このことによって、モデルがどう動作するのか、そのパラメータが学習でどう更新されるかを理解することが容易になる。また大量の学習データかオンライン学習機能が必要である。これは、問題解決のシミュレーションが必要であるか、実稼働データで学習する場合は学習中にＮＮ＋ＦＳシステムがおかしくなっても危険にならないことが求められることを意味する。

最後に一言、ＮＮ＋ＦＳシステムをいかなる問題にも対処可能な万能薬などと考えず、知的システムを設計する際の有用なツールとしてとらえることだ。

参考文献

[1] I. Aleksander, Helena Morton: An Introduction to Neural Computing. Chapman and Hall, (1990)

[2] A. G. Barto et al.: Neuronlike adaptive elements that can solve difficult learning control problems. IEEE Trans. Systems, Man & Cybernetics, 13, pp. 834–846, (1983)

[3] H. R. Berenji: A reinforcement learning-based architecture for fuzzy logic control. Int. J. Approximate Reasoning, 6, pp. 267 –292, February (1992)

[4] H. R. Berenji, P. Khedkar: Learning and tuning fuzzy logic controllers through reinforcements. IEEE Trans. Neural Networks, 3, pp. 724–740, September (1992)

[5] H. Bersini et al.: A simple direct adaptive fuzzy controller derived from its neural equivalent. In Proc. IEEE Int. Conf. on Fuzzy Systems 1993, pp. 345–350, San Francisco, March (1993)

[6] J. C. Bezdek, S. K. Pal (eds.): Fuzzy Models for Pattern Recognition. IEEE Press, (1992)

[7] K. Brahim, A. Zell: Anfis–snns: Adaptive network fuzzy inference system in the stuttgart neural network simulator. In R. Kruse et al. (eds.): Fuzzy Systems in Computer Science. Vieweg, (1994)

[8] G. A. Carpenter, S. Grossberg: Art2: Self–organization of stable category recognition codes for analog input patterns. Applied Optics, 26, pp. 4919–4930, (1987)

[9] D. Driankov et al.: An Introduction to Fuzzy Control. Springer–Verlag, (1993)

[10] P. Eklund: GeDeMeDeS a generic system for developing medical decision support. In Proc. First European Congress on Fuzzy and Intelligent Technologies (EUFIT93), pp. 801–803, Aachen, (1993)

[11] P. Eklund et al.: A generic neuro–fuzzy tool for developing medical decision support. In Patrik Eklund (ed.): Proc. of the MEPP'92, pp. 11–27, Åbo, (1992). Åbo Akademi Press

[12] P. Eklund, R. Fullér: A neuro–fuzzy approach to medical diagnostics. In Proc. First European Congress on Fuzzy and Intelligent Technologies (EUFIT93), pp. 810–813, Aachen, (1993)

[13] S. Fraleigh: Fuzzy logic and neural networks. PC AI, 8, 3, pp. 16–21, May/June (1994)

[14] J. Godjevac: State of the art in the neuro fuzzy field. Technical Report R93.25, LAMI–EPFL, Lausanne, (1993)

[15] P. Goode, Mo-yuen Chow: A hybrid fuzzy/neural system used to extract heuristic knowledge from a fault detection problem. In Proc. IEEE Int. Conf. on Fuzzy Systems 1994, pp. 1731–1736, Orlando, June (1994)

[16] S. K. Halgamuge et al.: Fuzzy - Neural Clustering Methods for Real - Time Classification. In European Congress on Fuzzy and Intelligent Technologies' 94, Aachen, Germany, September (1994)

[17] S. K. Halgamuge et al.: A Sub Bayasian Nearest Prototype Neural Network with Fuzzy Inte rpretability for Diagnosis Problems. In ACM Symposium on Applied Computing (SAC'95) (invited session), Nashville, USA, Februray (1995)

[18] S. K. Halgamuge et al.: An Alternative Approach for Generation of Membership Functions and Fuzzy Rules Based on Radial and Cubic Basis Function Networks. International Journal of Approximate Reasoning (in press), , , (1995)

[19] S. K. Halgamuge et al.: Fast perceptron learning by fuzzy controlled dynamic adaption of network parameters. In R. Kruse et al. (eds.): Fuzzy Systems in Computer Science, pp. 129–139. Vieweg, (1994)

[20] S.K. Halgamuge, M. Glesner: A fuzzy–neural approach for pattern classification with the generation of rules based on supervised learning. In Proc. Neuro–Nimes 92, Nimes, (1992)

[21] S.K. Halgamuge, M. Glesner: Neural networks in designing fuzzy systems for real world applications. Fuzzy Sets and Systems, 65, pp. 1–12, (1994)

[22] J.-S. R. Jang: Fuzzy modeling using generalized neural networks and Kalman filter algorithm. In Proc. of the Ninth National Conf. on Artificial Intelligence (AAAI-91), pp. 762–767, July (1991)

[23] J.-S. R. Jang: Rule extraction using generalized neural networks. In R. Lowen, M. Roubens (eds.): Proc. 4th IFSA Congress, volume Artificial Intelligence, pp. 82–85, July (1991)

[24] J.-S. R. Jang: ANFIS: Adaptive-network-based fuzzy inference systems. IEEE Trans. Systems, Man & Cybernetics, 23, pp. 665–685, (1993)

[25] E. A. Johnson, C.-H. Wu: A real–time fuzzy logic–based neural facial feature extraction technique. In Proc. IEEE Int. Conf. on Fuzzy Systems 1994, pp. 268–273, Orlando, (1994)

[26] E. Khan, P. Venkatapuram: Neufuz: Neural network based fuzzy logic design algorithms. In Proc. IEEE Int. Conf. on Fuzzy Systems 1993, pp. 647–654, San Francisco, March (1993)

[27] T. Kohonen: Self-organized formation of topologically correct feature maps. Biological Cybernetics, 43, pp. 59–69, (1982)

[28] B. Kosko: Neural Networks and Fuzzy Systems. A Dynamical Systems Approach to Machine Intelligence. Prentice-Hall, (1992)

[29] B. Krause et al.: A neuro–fuzzy adaptive control strategy for refuse incineration plants. Fuzzy Sets and Systems, 63, pp. 329–338, (1994)

[30] R. Kruse et al.: Foundations of Fuzzy Systems. Wiley, (1994)

[31] R. Kruse et al. (eds.): Fuzzy Systems in Computer Science. Vieweg, (1994)

[32] C. C. Lee: Fuzzy logic in control systems: Fuzzy logic controller, part i. IEEE Trans. Systems, Man & Cybernetics, 20, pp. 404–418, (1990)

[33] F. Mondada et al.: Mobile robot miniaturisation: A tool for investigation in control algorithms. In Proc. Third Int. Symposium on Experimental Robotics, Kyoto, (1993)

[34] H. Narazaki, Anca L. Ralescu: A synthesis method for multi-layered neural network using fuzzy sets. In IJCAI-91: Workshop on Fuzzy Logic in Artificial Intelligence, pp. 54–66, Sydney, August (1991)

[35] D. Nauck: A fuzzy perceptron as a generic model for neuro–fuzzy approaches. In Proc. Fuzzy–Systeme'94, Munich, October (1994)

[36] D. Nauck et al.: Combining neural networks and fuzzy controllers. In Erich Peter Klement, Wolfgang Slany (eds.): Fuzzy Logic in Artificial Intelligence (FLAI93), pp. 35–46, Berlin, (1993). Springer–Verlag

[37] D. Nauck et al.: Neuronale Netze und Fuzzy–Systeme. Vieweg, (1994)

[38] D. Nauck, R. Kruse: Interpreting changes in the fuzzy sets of a self-adaptive neural fuzzy controller. In Proc. Second Int. Workshop on Industrial Applications of Fuzzy Control and Intelligent Systems (IFIS'92), pp. 146–152, College Station, Texas, December (1992)

[39] D. Nauck, R. Kruse: A neural fuzzy controller learning by fuzzy error propagation. In Proc. Workshop of North American Fuzzy Information Processing Society (NAFIPS92), pp. 388–397, Puerto Vallarta, December (1992)

[40] D. Nauck, R. Kruse: A fuzzy neural network learning fuzzy control rules and membership functions by fuzzy error backpropagation. In Proc. IEEE Int. Conf. on Neural Networks 1993, pp. 1022–1027, San Francisco, March (1993)

[41] D. Nauck, R. Kruse: NEFCON–I: An X–Window based simulator for neural fuzzy controllers. In Proc. IEEE Int. Conf. Neural Networks 1994 at IEEE WCCI'94, pp. 1638–1643, Orlando, June (1994)

[42] H. Nomura et al.: A learning method of fuzzy inference rules by descent method. In Proc. IEEE Int. Conf. on Fuzzy Systems 1992, pp. 203–210, San Diego, (1992)

[43] A. Nowé, R. Vepa: A reinforcement learning algorithm based on 'safety'. In Erich Peter Klement, Wolfgang Slany (eds.): Fuzzy Logic in Artificial Intelligence (FLAI93), pp. 47–58, Berlin, (1993). Springer–Verlag

[44] W. Pedrycz, W. C. Card: Linguistic interpretation of self-organizing maps. In Proc. IEEE Int. Conf. on Fuzzy Systems 1992, pp. 371–378, San Diego, (1992)

[45] T. J. Procyk, E. H. Mamdani: A linguistic self–organizing process controller. Automatica, 15, pp. 15–30, (1979)

[46] W. Z. Qiao et al.: A rule self-regulating fuzzy controller. Fuzzy Sets and Systems, 47, pp. 13–21, (1992)

[47] A. Rueda, W. Pedrycz: A hierachical fuzzy–neural–pd controller for robot manipulators. In Proc. IEEE Int. Conf. on Fuzzy Systems 1994, pp. 672–677, Orlando, (1994)

[48] David E. Rumelhart, James L. McClelland (eds.): Parallel Distributed Processing: Explorations in the Microstructures of Cognition. Foundations, volume 1. MIT Press, (1986)

[49] H. Schreiber, S. Heine: Einsatz von Neuro–Fuzzy–Technologien für die Prognose des Elektroenergieverbrauches an "besonderen" Tagen. In Proc. 4. Dortmunder Fuzzy–Tage, Dortmund, (1994)

[50] S. Shao: Fuzzy self-organizing controller and its application for dynamic processes. Fuzzy Sets and Systems, 26, pp. 151–164, (1988)

[51] H. Steinmüller, O. Wick: Fuzzy and neuro fuzzy applications in european wahing machines. In Proc. First European Congress on Fuzzy and Intelligent Technologies (EUFIT93), pp. 1031–1035, Aachen, (1993)

[52] S. M. Sulzberger et al.: Fun: Optimization of fuzzy rule based systems using neural networks. In Proc. IEEE Int. Conf. on Neural Networks 1993, pp. 312–316, San Francisco, March (1993)

[53] H. Takagi, I. Hayashi: NN–driven fuzzy reasoning. Int. J. Approximate Reasoning, 5, pp. 191–212, (1991)

[54] C. von Altrock et al.: Advanced fuzzy logic control technologies in automotive applications. In Proc. IEEE Int. Conf. on Fuzzy Systems 1992, pp. 835–842, San Diego, (1992)

[55] D. A. White, Donald A. Sofge (eds.): Handbook of Intelligent Control. Neural, Fuzzy, and Adaptive Approaches. Van Nostrand Reinhold, (1992)

[56] G. Wöhlke: A neuro–fuzzy based system architecture for the intelligent control of multi–finger robot hands. In Proc. IEEE Int. Conf. on Fuzzy Systems 1994, pp. 64–69, Orlando, (1994)

[57] A. Zell et al.: SNNS User Manual 3.0. University of Stuttgart, August (1992)

[58] J. M. Zurada: Introduction to Artificial Neural Systems. West Publishing Company, (1992)