

Online Neuro-Fuzzy CANFIS Hidden-Node Teaching

Eiji Mizutani

Department of Industrial Management
National Taiwan University of Science and Technology
Taipei, Taiwan
Email: eiji@mail.ntust.edu.tw

Jing-Yun Carey Fan

Coach Team/Mobile Division
Zoran Corporation
Hsinchu, Taiwan
Email: goldfish@gmail.com

Abstract—On-line first-order backpropagation (BP) has been widely employed for optimizing a multi-layer neural network and a fuzzy neural network. When BP is applied to a TSK fuzzy system, the interpretability of fuzzy rules may be lost. We describe a simple and effective remedy for the loss by casting the posed optimization problem into a general Bolza-type optimal-control mold so as to include stage costs on top of the terminal cost; this is what we called the (on-line) hidden-node teaching. Our on-line learning scheme turns out to be useful not only in optimizing a large model, a so-called CANFIS neuro-fuzzy modular network, but also in enhancing its generalization capacity. The fundamental concept is that each (local) expert network is supervised individually by hidden-node teaching for minimizing a stage cost while all local-expert modules are encouraged to cooperate in reducing the terminal cost simultaneously. Furthermore, we show how to construct a CANFIS modular network in order to alleviate the so-called curse of dimensionality that frequently hampers the design of fuzzy systems. In simulation, our concepts have been demonstrated in the letter recognition benchmark problem as well as in small regression and XOR-classification tasks.

Keywords—CANFIS modular networks; hidden-node teaching.

I. INTRODUCTION

Consider a typical single-output fuzzy model with a given set of M fuzzy “if-then” rules; its final output y may be produced as

$$y = \sum_{p=1}^M w_p h_p, \quad (1)$$

where h_p , each of *then*-part outputs, is weighted by w_p , the firing strength of rule p , which is determined by *if*-part of fuzzy membership functions (MFs). When each h_p is expressed as a *linear* system, as shown next, the preceding fuzzy model is a well-known TSK (Takagi-Sugeno-Kang) fuzzy system:

$$h_p = \theta_{0,p} + \sum_{i=1}^S x_i \theta_{i,p}, \quad (2)$$

where S is the dimension of the input vector $\mathbf{x}$, and $\theta_{i,p}$ (with bias $\theta_{0,p}$) are the linear parameters. Since TSK was developed in 1980s [22], [21], it has enjoyed great popularity in diverse contexts, probably due to its simple linear *then*-part structure.

Our motivating example is a simple data fitting (regression) task described for Fig.1 on p.118 in [22], a task described for the original Takagi-Sugeno fuzzy system. In our interpretation, linear *then*-parts are given as h_p in the form of Equation (2), where $p=1, 2$, for the following two fuzzy rules: (Rule 1) if input x is “small,” then use linear h_1 , and (Rule 2) if input x is “large,” then use linear h_2 . Those two fuzzy rules cooperate in

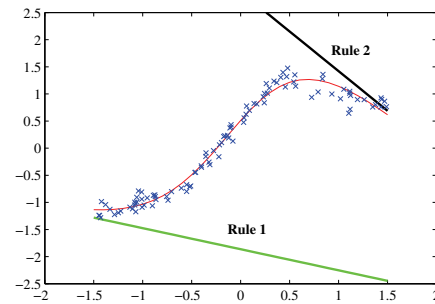


Fig. 1. An after-learning result obtained by “standard” on-line first-order BP on a TSK fuzzy system with two rules in a regression task that involves 100 data (marked by “x”), adapted from p.118 in [22]. The *then*-part linear rule’s output deviates significantly from the data distribution in an unexpected manner.

producing final responses; this is what we call *complementary learning*. When we attack the posed regression task in conjunction with the “standard” on-line first-order backpropagation (BP), a problem often arises that h_p , the *then*-part of Rule p , may deviate from the data distribution beyond our expectation (hence, the loss of interpretability); Figure 1 shows such a situation. In the original Takagi-Sugeno fuzzy system [22], given *if*-part MF setups, RLSE (recursive least-squares estimation), implemented by the well-known Sherman-Morrison inverse formula, is applied in “batch” mode to optimizing *then*-part *linear* parameters. (The same method is used in ANFIS hybrid-learning.) Such a batch-mode RLSE-based approach may yield more interpretable *then*-part results than on-line first-order BP, but its batch-mode use of the Sherman-Morrison formula is not recommendable (see [8], [7]). More important, direct RLSE is not applicable to “nonlinear” *then*-part that can overcome the so-called curse of dimensionality when S , the dimension of inputs, is large. Later in Sec.III-B, we demonstrate our approach to the letter recognition benchmark problem ($S=16$).

In what follows, we describe a Bolza-type optimal-control formulation that can resolve the preceding problem indicated in Figure 1. We then discuss how to mitigate the curse of dimensionality, which becomes a serious obstacle in the design of fuzzy systems when the number of inputs increases. Throughout this paper, we highlight CANFIS (Co-Active Neuro-Fuzzy Inference System) [5], for its systematic design methodology offers a useful and consistent framework for describing representative adaptive-network models; such as

an aforementioned TSK, MLP (multilayer perceptron), RBFN (radial basis function network), and modular networks. In all numerical tests that follow, the bell-shaped Gaussian MF is used for the if-part of each fuzzy rule, and its center and width parameters are fine-tuned during the learning phase.

II. ON-LINE CANFIS HIDDEN-NODE LEARNING

The first-order backpropagation (BP) is a simplified form of the Kelley-Bryson optimal-control gradient procedure (see [6] and references therein); it is originally contrived to attack a general Bolza-type optimal-control problem. We explain how such a general problem formulation inspires an elaborate neuro-fuzzy learning algorithm.

A. The Bolza-type optimal-control formulation

We describe an N -stage (discrete-time) optimal-control problem of Bolza in the framework of N -layer MLP-learning, where stage s (i.e., layer s) has P_s states (or nodes), and stage 1 corresponds to the first (nominal) input layer. The objective function J below includes an intermediate *stage cost* L^s at the hidden layer s as well as the *terminal cost* E at layer N (last stage):

$$J(\theta) = \sum_{s=2}^{N-1} L^s(\mathbf{y}^s, \theta^s) + E(\mathbf{y}^N), \quad (\text{with } \mathbf{y}^1 \text{ given}) \quad (3)$$

where $E(\cdot)$, the terminal cost at stage N , depends only on terminal *state* vector $\mathbf{y}^N$ of length P_N (per datum), and $L^s(\cdot, \cdot)$, the cost at stage s , depends on both the P_s -dimensional *state* vector $\mathbf{y}^s$ of “after-node” outputs and the n_s -dimensional *control* vector θ^s of weight parameters (including thresholds); hence, the length $n_s \equiv (1 + P_s)P_{s+1}$ in MLP-learning. Equation (3) may consist of squared residuals on each (training) datum d (for $d=1, \dots, D$) at each stage s , where each residual vector $\mathbf{r}_d^s$ (of length P_s) is evaluated as the difference between the node-output vector $\mathbf{y}_d^s$ and the target vector $\mathbf{t}_d^s$. In fact, it is often the case that the terminal cost is defined as the squared-error measure below:

$$\begin{aligned} E(\mathbf{y}^N) &= \frac{1}{2} \sum_{d=1}^D \sum_{k=1}^{P_N} (y_{k,d}^N - t_{k,d}^N)^2 \\ &= \frac{1}{2} \sum_{d=1}^D \|\mathbf{y}_d^N - \mathbf{t}_d^N\|_2^2 = \frac{1}{2} \sum_{d=1}^D \mathbf{r}_d^{N\top} \mathbf{r}_d^N. \end{aligned} \quad (4)$$

Likewise, the stage cost $L^s(\cdot, \cdot)$ ($1 < s < N$) can be defined as

$$\begin{aligned} L^s(\mathbf{y}^s, \theta^s) &= \frac{1}{2} \sum_{d=1}^D \sum_{k=1}^{P_s} (y_{k,d}^s - t_{k,d}^s)^2 + \frac{1}{2} \sum_{j=1}^{n_s} (\theta_j^s)^2 \\ &= \frac{1}{2} \sum_{d=1}^D \mathbf{r}_d^{s\top} \mathbf{r}_d^s + \frac{1}{2} \theta^{s\top} \theta^s, \end{aligned} \quad (5)$$

where the last quantity is a so-called weight-decay term for penalizing large weights in modulus, and $\mathbf{r}_d^s$ is the P_s -length vector of hidden residuals on datum d evaluated at intermediate layer s ($1 < s < N$). The introduction of such a stage

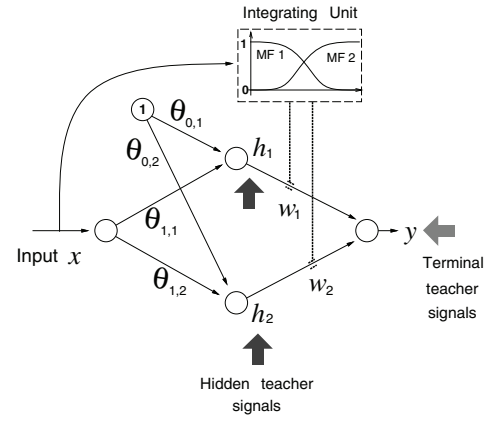


Fig. 2. A 1-2-1 MLP representation of CANFIS with two linear rules, in which two terminal weights, w_1 and w_2 , are the firing strengths of corresponding two rules, Rule 1 and Rule 2, respectively, obtainable from the fuzzy membership functions. The posed network is equivalent to a TSK fuzzy system (see p.377 [5]). For hidden-node teaching, teacher signals are presented to the hidden layer as well as to the terminal output layer. In this model, each of two local-expert is a linear perceptron; expanding each of them to an MLP leads to a CANFIS modular network (see Figure 4).

cost $L^s(\cdot, \cdot)$ at intermediate stages (i.e., at *hidden layers*; $1 < s < N$) is what we call *hidden-node teaching* [6], [9], [13]. For MLP-type node dynamics at layer s , $\mathbf{y}^s \equiv \mathbf{f}(\mathbf{x}^s, \theta^s)$, we define two types of *node sensitivities*: (1) after-node sensitivity: $\xi^s \equiv \frac{\partial J}{\partial \mathbf{y}^s}$ and (2) before-node sensitivity: $\delta^s \equiv \frac{\partial J}{\partial \mathbf{x}^s}$; the former is used (as state sensitivity) in optimal control, whereas the latter is well-known for a generalized delta rule in the neural-network literature (e.g., see [6], [13]). Then, the Kelley-Bryson optimal-control gradient formula (on datum d with subscript d suppressed) is given by

$$\underbrace{\xi^s}_{P_s \times 1} = \Theta_{\text{void}}^{s\top} \delta^{s+1} + (\mathbf{y}^s - \mathbf{t}^s), \quad (6)$$

where Θ_{void}^s denotes a $P_{s+1} \times P_s$ matrix of parameters at stage s excluding “threshold” parameters (if any at stage s). By the above formula, the node sensitivities are propagated backwards in a layer-by-layer (i.e., stagewise) fashion.

B. CANFIS with linear rules

As shown in Equation (2), the *then*-part of TSK fuzzy rules is identical to a “linear” *perceptron*. Given M fuzzy rules, TSK combines (at most) M perceptrons in a fuzzy fashion on each datum. This can be organized into our familiar MLP-type architecture. In this perspective, a single-output TSK can be expressed as an S - M -1 MLP with M “linear” *hidden* nodes (hence, $P_1 = S$; $P_2 = M$; $P_3 = 1$). Each hidden-node output is given by h_p , as defined in Equation (2), where the parameters $\theta_{i,p}$ are the weights between the input layer and hidden layer (with bias $\theta_{0,p}$). The weights (with no bias) between the terminal and hidden layers correspond to firing strengths w_p , which are *data-dependent* values.

Example: A single-input single-output TSK with two rules. A TSK with two fuzzy rules ($M = 2$) can be constructed as a CANFIS neuro-fuzzy network with two perceptrons since

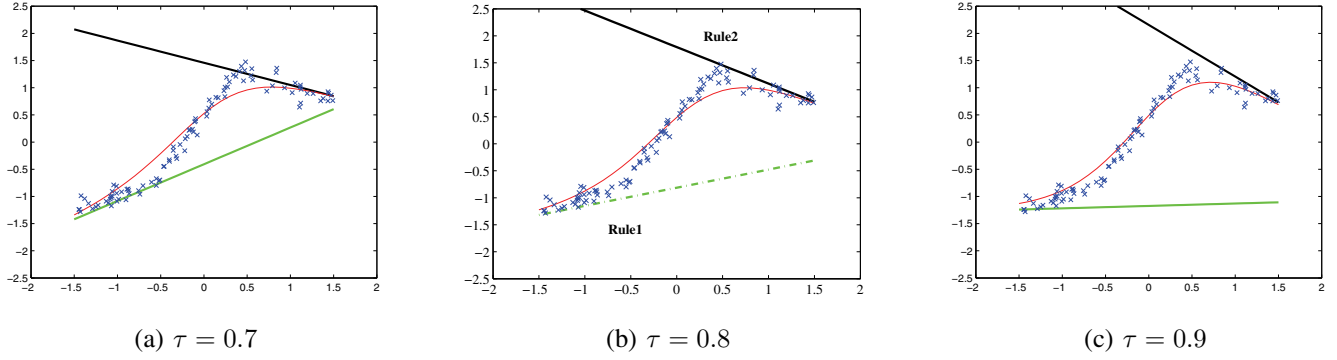


Fig. 3. Data fitting results obtained by on-line hidden-node teaching with three choices of τ (for designating the local-expert region) in Equation (8): (a) $\tau = 0.7$; (b) $\tau = 0.8$; and (c) $\tau = 0.9$. The results indicate that $\tau \approx 0.7$ would be a good choice in the posed problem. Compared with Figure 1, where no hidden-node teaching is applied, it is clear that hidden-node teaching is effective in maintaining rules' *then*-part interpretability to meaningful limits. In each case, all the parameters in both *if*-part and *then*-part are updated on-line until the same error level is achieved.

it can be put in a three-layer ($N = 3$) 1-2-1 MLP mold having single-input and single-output nodes ($P_1 = P_3 = 1$) with two *linear* hidden-node functions ($P_2 = 2$), as illustrated in Figure 2: Here, two terminal weights, denoted by w_1 and w_2 , are the *firing strengths* calculated at the integrating unit of fuzzy MFs in response to each input x , and two hidden-node “linear” outputs h_i correspond to *then*-part of fuzzy rules; that is, Equation (2) reduces to

$$h_i = \theta_{0,i} + x\theta_{1,i} \quad \text{for } i = 1, 2. \quad (7)$$

The final output y is generated by $y = w_1 h_1 + w_2 h_2$, as shown in Equation (1) with $M = 2$. $\square$

The above transformation of a fuzzy system to such an MLP-type network as in Figure 2 is a key ingredient in design for CANFIS (see p.371 [5] for more details).

C. TSK hidden-node teaching effects in a motivating example

On-line first-order BP-based algorithms may be attractive in attacking a large-scale problem. However, when on-line BP is applied to optimizing both *if*-part and *then*-part of fuzzy rules, after-learning knowledge extraction may not always be possible. Here, for the initial setup of fuzzy rules, the *then*-part *linear* parameters are randomly initialized in a small range, as usually done for MLP-learning. In order to attain a higher input-output mapping precision, on-line BP may adapt the parameters of initial fuzzy rules significantly, ending up with meaningless fuzzy rules. This is a dilemma between mapping precision and interpretability (see pp.382–387 [5]). Back to our motivating example, Figure 1 illustrates such a dilemma; this phenomenon is partly due to a fact that the standard supervised learning uses a terminal cost alone, as in Equation (4), and thus only encourages the final combined output to be close to the desired output; this is the nature of *complementary* learning. In the spirit of fuzzy systems, however, the *then*-part linear output of each rule should match up to the desired output when its firing strength is large enough; this is the original claim made for a TSK (see p.118 in [22]).

Hidden-node teaching can handle this situation with a CANFIS network depicted in Figure 2. The posed two-rule

CANFIS has two terminal weights: $\Theta_{\text{void}} \equiv [w_1, w_2]$. Now, suppose that a “terminal” teacher signal, denoted by t^3 , is presented to the terminal node at layer 3 and another “hidden” teacher signal, t_i^2 , to hidden-node i at layer 2; then, the scalar version of the on-line hidden-node teaching BP-formula (6) is given by

$$\xi_i^2 = w_i(y_1^3 - t^3) + (y_i^2 - t_i^2) = w_i(y - t^3) + (h_i - t_i^2),$$

where $\xi^3 = \delta^3 = (y - t^3)$; y_k^s is the k th node output at layer s ; and the last equality holds owing to Equations (1) and (2). In our numerical experiments that follow, we introduce scalar τ as a threshold for designating local-expert regions, and the same desired outputs t are presented at both terminal and hidden layers in our model, yielding the following BP-formula (with superscripts omitted for simplicity):

$$\xi_i = \begin{cases} w_i(y - t) + (h_i - t), & \text{if } w_i \geq \tau \\ w_i(y - t), & \text{otherwise.} \end{cases} \quad (8)$$

Since the hidden-node functions are also *linear*, as shown in Equation (7), we have $\delta_i = \xi_i$ at hidden node i . We thus obtain the gradient vector below for *then*-part parameters θ in response to input x :

$$\frac{\partial J}{\partial \theta} = \left[\frac{\partial J}{\partial \theta_{0,1}}, \frac{\partial J}{\partial \theta_{1,1}}, \frac{\partial J}{\partial \theta_{0,2}}, \frac{\partial J}{\partial \theta_{1,2}} \right]^T = [\delta_1, x\delta_1, \delta_2, x\delta_2]^T,$$

for which the weight-decay term is omitted from Equation (5). Once a (scalar) gradient element $g = \frac{\partial J}{\partial \theta}$ is evaluated for any parameter θ , it is used to update θ on-line by

$$\begin{cases} \Delta \theta = -\eta g + \beta \Delta \theta_{\text{old}}, \\ \theta_{\text{new}} = \theta_{\text{old}} + \Delta \theta, \\ \Delta \theta_{\text{old}} \leftarrow \Delta \theta, \end{cases} \quad (9)$$

where η is a learning rate and β a momentum term.

Figure 3 shows our numerical demonstration with three different choices of τ for the hidden-node teaching BP-formula (8). The larger the value τ is, the smaller each local-expert region is. When $\tau = 1$ in Equation (8), essentially no hidden-node teaching applies. This preliminary investigation

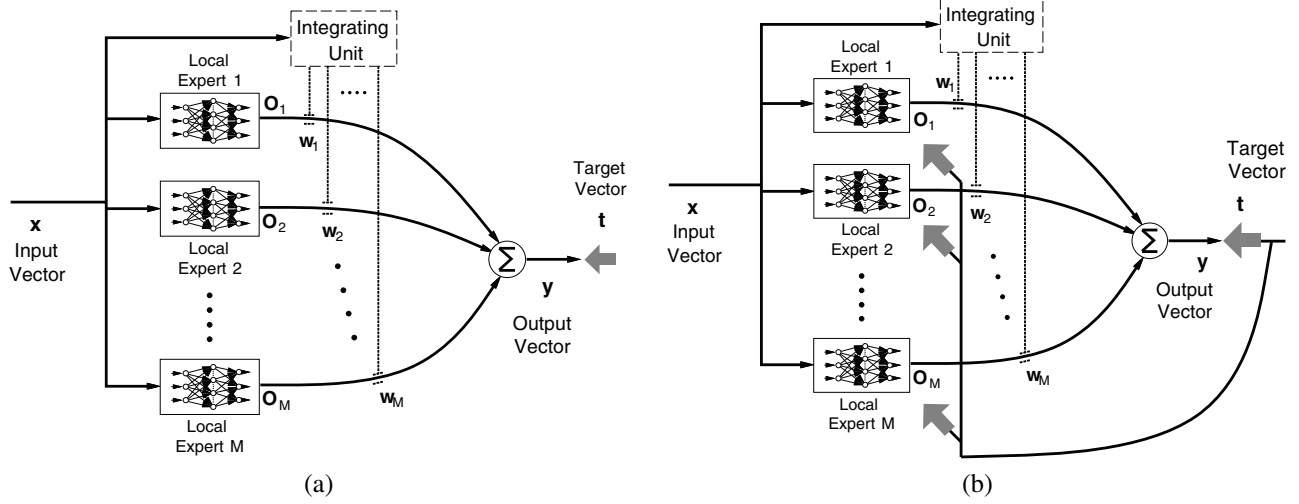


Fig. 4. An architectural comparison between CANFIS modular networks (a) without and (b) with hidden-node teaching [see Eq.(11)]. Here, *then*-part local-expert MLPs are encouraged to cooperate with each other within overlapped regions, where their corresponding *if*-part fuzzy MFs are overlapped in the input space.

suggests $\tau \approx 0.7$. Compared with Figure 1, the *then*-part outputs are kept within the data-distribution limits. In this way, hidden-node teaching can maintain fuzzy rules' interpretability.

D. CANFIS modular networks

Next, we advance to “non-linear” *then*-part of fuzzy rules that can circumvent many problems arising in “linear” *then*-part fuzzy rules. In Equation (7), each *then*-part h_i is a simple “linear” perceptron (with only two weights in a single-input case). If each h_i is expanded to include any number of “hidden” layers (with nonlinear hidden-node functions), then each amounts to an MLP (see p.377 [5] for such a transformation); the resulting model is a CANFIS with “neural” rules; or, a CANFIS modular network with local-expert MLPs. Figure 4 describes a typical CANFIS modular network, in which the final combined output is given by

$$\mathbf{y} = \sum_{i=1}^M w_i \mathbf{o}_i, \quad (10)$$

where w_i is the firing strength of Rule i , and $\mathbf{o}_i$ the output of *then*-part local-expert MLP i . To emphasize the usefulness of the posed model, we later show how to approach the UCI letter recognition benchmark (26-class pattern classification) problem that involves with 16 input features.

As described earlier for Figure 1, in standard supervised learning with terminal residuals in Equation (4) alone, the output of each *then*-part expert-MLP may differ substantially from the desired output even in its own local-expert region (e.g., see Fig.13.15 on p.391 [5]) due to the nature of *complementary learning*. That is, when a modular network (that consists of M local-expert MLPs) is optimized simultaneously, only the final outcome $\mathbf{y}$ is made close to the terminal desired output $\mathbf{t}$; here, each local-expert output $\mathbf{o}_i$ may not reproduce the terminal desired output $\mathbf{t}$ accurately even when the input $\mathbf{x}$ comes right into its own territory. In the spirit of fuzzy systems, it may be adequate to encourage each expert to generate the terminal

desired outputs when the input $\mathbf{x}$ falls into its own local-expert area. For this reason, hidden-node teaching may apply to a CANFIS modular network, as shown in Figure 4(b).

By hidden-node teaching, we may designate a “principal” local-expert MLP based on the magnitude of firing strengths w_i . From the concept behind Equation (8), we say that on each datum, local-expert MLP i is a principal expert if its firing strength w_i is greater than or equal to a threshold τ . By hidden-node teaching, the principal local-expert MLP is encouraged to reproduce the same target $\mathbf{t}$ as presented at the terminal layer. To this end, Equation (8) is modified as

$$\xi_i = \begin{cases} w_i(\mathbf{y} - \mathbf{t}) + (\mathbf{o}_i - \mathbf{t}), & \text{if } w_i \geq \tau \\ w_i(\mathbf{y} - \mathbf{t}), & \text{otherwise} \end{cases}, \quad (11)$$

where term $(\mathbf{o}_i - \mathbf{t})$ is a “hidden” residual vector for each expert-MLP i evaluated at its final output layer [see Fig.4(b)].

We summarize a general computing procedure below for CANFIS modular-network learning with hidden-node teaching using the same symbols used in Figure 4:

Algorithm for CANFIS with M local-expert MLPs in Fig.4:

- (1) Do forward pass in the integrating unit of fuzzy (Gaussian) MFs to calculate M rules' firing strengths w_i (typically by the product rule).
- (2) Do forward pass in each local-expert MLP i to evaluate MLP's output vector $\mathbf{o}_i$, the output of *then*-part of Rule i , (when $w_i > 0$) for $i = 1, \dots, M$.
- (3) Evaluate the final CANFIS output $\mathbf{y} = \sum_{i=1}^M w_i \mathbf{o}_i$, and then compute the terminal residual vector $\mathbf{r} = \mathbf{y} - \mathbf{t}$.
- (4) For each local-expert MLP i ($i = 1, \dots, M$), repeat the following backward-pass procedure:
 - (a) Propagate back the terminal after-node sensitivity vector, residual vector $\mathbf{r}$, to local-expert MLP i .
 - (b) If Rule i is fired strongly enough [i.e., $w_i \geq \tau$ in Eq.(11); see also Eq.(8)], then compute the hidden

residual vector and add it to the back-propagated after-node sensitivity signals.

- (c) Continue the backward pass for the remaining stages to update the weights.

(5) Do backward pass on the integrating unit of fuzzy MFs. $\square$

General parameter-updating formulas for use in Steps (4) and (5) can be found on p.379 [5]. In Step (5), MF-parameters may be updated on-line (on each datum); alternatively, the gradient information of MF-parameters may be accumulated over all the training data for the “batch-mode” fine-tunings (see [4], [10]), while the other parameters of local-expert MLPs are updated on-line.

Furthermore, a so-called *truncation filter* function (see p.380 [5]) may be introduced; for instance, in attacking classification problems with binary target values, the k th residual at node k is set equal to zero for any final output y_k beyond the target value t_k , *ON* (+1) or *off* (−1); this implies the following residual evaluation:

$$r_k = \begin{cases} 0 & \text{if } t_k = \text{ON and } y_k \geq \text{ON} \\ 0 & \text{if } t_k = \text{off and } y_k \leq \text{off} \\ y_k - t_k & \text{otherwise.} \end{cases} \quad (12)$$

Here, the k th element of final outputs y [see Equation (10) and Figure 4(b)] is given by

$$y_k = \sum_{i=1}^M w_i o_{i,k},$$

when M expert-MLPs are employed. The posed scheme is related to a threshold least-mean-square procedure (e.g., see [20]). If “hidden” target signals at the output layer of each expert-MLP are the same as the “terminal” target ones for hidden-node teaching [see Fig.4(b)], then those associated hidden residuals may also be subject to Equation (12), for which y_k is replaced with $o_{i,k}$, the k th output of expert-MLP i .

III. NUMERICAL EXPERIMENTS

We explain more about the learning capacity of CANFIS models, showing numerical evidence obtained in the well-known XOR (two-class pattern classification) problem and the UCI letter recognition benchmark problem.

A. The XOR problem by CANFIS with two linear rules

In the literature, the XOR problem is usually solved by a 2-2-1 MLP; yet, when the so-called weight-sharing is applied to hidden weights, it can be solved by a 1-2-1 MLP with $x_1 + x_2$ as input. The solution weights of the 1-2-1 MLP can be always expanded as those of a corresponding 2-2-1 MLP (see [14] for details). Similarly, for attacking the XOR problem, a conventional fuzzy system usually needs four rules: two MFs on each input x_i ($i = 1, 2$). In this section, we demonstrate a 1-2-1 MLP-structured CANFIS in Figure 2 for solving the XOR problem. By the weight-sharing concept, the original four XOR-data reduce to three data, as shown in the right-most column in Table I. In “standard” MLP-learning, the hidden-node activation function should be nonlinear. For

TABLE I
FOUR DATA OF THE XOR, TWO-BIT PARITY, PROBLEM: (COLUMN 1) TARGET VALUE, t ; (COLUMNS 2 AND 3) TWO-BIT INPUTS, x_1 AND x_2 ; AND (COLUMN 4) $x_1 + x_2$ FOR THE WEIGHT-SHARING CASE.

Target t	Two inputs		Sum of inputs $x_1 + x_2$
	x_1	x_2	
−1	+1	+1	+2
+1	−1	+1	0
+1	+1	−1	0
−1	−1	−1	−2

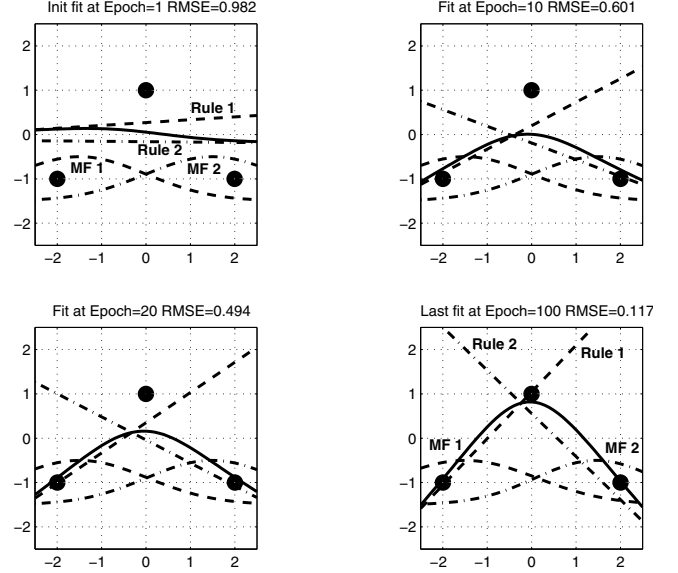


Fig. 5. A learning progress of a CANFIS with two linear rules, a 1-2-1 MLP-structured CANFIS in Figure 2, for solving the XOR problem: Displayed are the terminal CANFIS output (solid line) as well as two *if*-part MF outputs & two *then*-part linear outputs at epochs 1 (upper left), 10 (upper right), 20 (lower left), and 100 (lower right). The three dots denote the three data points, resulting from our weight-sharing; see the right-most column in Table I. The CANFIS is trained by on-line hidden-node teaching with $\tau = 0.95$ in Equation (8).

instance, an MLP with any number of “linear” hidden nodes is unable to solve the XOR problem because a linear transformation of another linear transformation is just *linear*. One may wonder why a CANFIS with two *linear* hidden nodes has a “nonlinear” mapping capacity. This is simply because the terminal weights w_i are firing strengths, whose values are data-dependent (see Figure 2).

Figure 5 shows four snapshots of the CANFIS output during on-line learning. As the number of epochs increases, the output approaches the three data points. In this way, it can be readily confirmed that the CANFIS with only two *linear* rules solves the XOR problem perfectly. The 1-2-1 MLP-structured CANFIS (in Figure 2) even with (two) *linear* hidden nodes has a great nonlinear mapping capacity, which is derived from the *data-dependent* terminal weights, firing strengths of two linear rules. By contrast, in the standard MLP-learning, terminal weights are *common* to all data points (hence, an MLP with “linear” hidden nodes would never solve XOR). This is a fundamental distinction between a MLP-structured CANFIS

and a standard MLP.

In the XOR or higher-dimensional parity problems, a standard MLP (with sigmoidal-shaped hidden-node functions) tends to fail due to hidden-node saturations. Originally, our hidden-node teaching scheme has been designed to alleviate such malicious hidden-node saturation problems for MLP-learning [9], although saturations may be necessary to some extent for parity solutions (see [14] for proofs). In seven-bit parity, for instance, a 7-4-1 MLP (with only four $\tanh$ hidden nodes) can develop *insensitivity* to initial weights owing to hidden-node teaching; that is, it can always solve the 7-bit parity problem (128 data) * “perfectly.”

The above *weight-sharing* concept is applicable to any number of (input) bits of parity; in 21-bit parity, for instance, the weight-sharing concept leads to a 1-11-1-MLP-structured CANFIS with 11 linear rules; this is because 21-bit parity can be readily solved by a standard 1-11-1 MLP, which yields a solution weight vector for a 21-11-1 MLP (see [14]). By contraries, the standard TSK fuzzy system needs to use at least 2^{21} rules since at least two fuzzy MFs, the minimum number of MFs, must be set up on each input feature. This is the so-called “curse of dimensionality” problem, which commonly arises in the design of many other fuzzy systems also. Unfortunately, the weight-sharing concept may not apply in general (see the letter recognition problem). In such a case, CANFIS modular networks may be useful in bypassing the curse of dimensionality through the subset selection of input features (see other applications in [4], [10]).

B. The UCI letter recognition benchmark problem

Next, we attack a larger and more realistic letter recognition (26-class pattern classification) benchmark problem available at the UCI machine learning repository [1]. The problem contains 16,000 patterns for training and 4,000 for test; each pattern consists of 16 extracted attributes (inputs) and 26 capital letters, English alphabet: A to Z (of 20 different fonts); hence, 26 classes in total. As a preprocessing, the training data set is normalized to *zero mean* and *unit variance* for each attribute. Accordingly, the test data set is preprocessed using the average of the training data set; here, we did not use any information of the average of test data. A straightforward application of TSK-type fuzzy system may end up with 2^{16} *then-part* linear functions (i.e., linear local-expert perceptrons in Figure 2) when two MFs, the minimum number of MFs, are designed on each input feature. This is the aforementioned “curse of dimensionality” problem, which often occurs in many fuzzy systems. To circumvent this dimensionality obstacle, we apply fuzzy partitioning only to a subset of inputs (see [4], [10]) and use local-expert MLPs as nonlinear *then-part* of fuzzy rules.

In the UCI-letter recognition problem, the 12th input (x_{12}) alone may be partitioned in a fuzzy fashion; e.g., with two “intuitively-obvious” linguistic labels on it: *small* and *large*.

*The Matlab code for 7-bit parity attacked by a 7-4-1 MLP is available at www.ieor.berkeley.edu/People/Faculty/dreyfus-pubs/hidteach.m for reference.

TABLE II

THE ERROR-RATE RESULTS IN THE UCI-LETTER RECOGNITION, AVERAGED OVER TEN TRIALS, OBTAINED AT (PRE-SET LIMIT) EPOCH 500 BY CANFIS HAVING TWO 16-70-50-26 (LOCAL EXPERT) MLPs, (IN COLUMNS 2 AND 3) WITH HIDDEN-NODE TEACHING ($\tau = 0.7$) AND (IN COLUMNS 4 AND 5) WITHOUT IT, IN CONJUNCTION WITH THE TRUNCATED FILTER FUNCTION FOR BOTH HIDDEN AND TERMINAL RESIDUALS. THE LAST ROW SHOWS THE ERROR RATE OBTAINED BY A SIMPLE COMMITTEE METHOD THAT EMPLOYED TEN TRAINED CANFIS MODELS.

	With hid-teach		No hid-teach	
	Train	Test	Train	Test
Average error (%)	0.19	3.43	0.35	3.42
Committee error (%)	0.13	2.35	0.25	2.43

This is because the distribution of x_{12} appeared to spread relatively widely, but our “improvisatory” choice of a single input feature x_{12} is purely for our testing purpose of CANFIS modeling to the letter recognition problem. In our experiment, we employ a CANFIS modular network with two local experts ($M = 2$), each of which is a two-hidden-layer 16-70-50-26 MLP (with 6,066 weights), the same-size MLP architecture used in [19], [12] with $\tanh$ hidden-activation and *linear*-output nodes. The posed CANFIS is optimized by *on-line* first-order BP (with or without hidden-node teaching) using momentum term 0.8 [$\beta = 0.8$ in Eq.(9)] and learning rates: $\eta_{h1} = 1 \times 10^{-4}$ for the hidden weights at Stage 1, $\eta_{h2} = 8 \times 10^{-5}$ for those at Stage 2, $\eta_o = 5 \times 10^{-5}$ for terminal weights. The integrating unit consists of two Gaussian MFs having the initial center positions -1.5 and 1.5 with the same width 1.0 initially. Those MFs are updated in batch-mode with a very small learning rate 1×10^{-8} merely for the fine-tuning purpose, whereas the two local-expert MLPs are trained in on-line mode with the above learning rates.

Table II shows the results obtained (at the pre-set limit epoch 500) by a CANFIS modular network with two local-expert 16-70-50-26 MLPs with or without hidden-node teaching. The last row shows the results obtained by the simple committee method (for model-averaging; see Ch.8 in [2]) using ten weight-value sets obtained at final (i.e., pre-set limit) epoch; each of those resulted from the ensemble over ten optimized CANFIS models. In general, hidden-node teaching may serve as a constraint, slowing down the learning process and resulting in a lower mapping precision. In this application, however, the hidden-node teaching results allude to an improvement in speed and accuracy with respect to training error rates; this may be due to increased “hidden” residuals (in addition to terminal residuals).

IV. CONCLUDING REMARKS AND FUTURE DIRECTIONS

The CANFIS design methodology may be regarded most simply as a way of converting an original TSK model into an MLP-type structure, but it can be readily extended to a more sophisticated CANFIS modular network (see p.377 [5]). This perspective makes CANFIS conceptually differ greatly from other network-based fuzzy models including a fuzzy neural network (e.g., see [3]). A key concept behind CANFIS modular-network learning is that, by selecting an appropriate

subset of input features, we can circumvent the so-called curse of dimensionality that frequently arises in any fuzzy systems. Since each local-expert MLP has a great learning capacity, any choice of small input subsets to the integrating unit may not deteriorate the overall learning performance. In the preceding application examples, we have designed CANFIS models by choosing only one feature for fuzzy partitioning, but our numerical test shows encouraging results (see also [4], [10] for other applications).

Yet another advantage of our CANFIS design is that all the parameters of any-size CANFIS modular network can be readily optimized by what we have developed, the (on-line) hidden-node teaching backpropagation algorithm, which exploits its multi-stage structure. Our hidden-node teaching method, which was originally developed to avoid hidden-node saturations in MLP-learning [9], may be significant for neuro-fuzzy learning for several different reasons: (1) for attacking a large-scale problem due to its easy on-line implementation; (2) for maintaining rules' interpretability in the spirit of a fuzzy system; (3) for improving generalization capacity; (4) for escaping from plateaus in learning. Merit (1) is obvious, and (2) was explained for Figure 3. For (3), in our experiments; the same target signals were presented at the hidden layer [i.e., the final output layer of each expert-MLP; see Figure 4(b)] as well as at the terminal layer of a CANFIS network. In consequence, the output variance of local experts could be reduced. Furthermore, the variance of linear weights may be decreased as well, compared with some other (neural) fuzzy systems that employ RLSE to determine the linear weights, as mentioned in Section I, because such a least squares estimator tends to yield quite large linear weights in magnitude; this is an important issue common to *separable nonlinear least squares* learning algorithms (e.g., see [17]). In any event, smaller variance in either output or weight or in both would furnish a better performance in response to previously unseen (test) data. Lastly, item (4) is new (our future work): The problem is that if hidden-node outputs get very similar to each other on all (training) data during learning, then a matrix of hidden-node outputs becomes *rank-deficient*, giving rise to plateaus in learning (e.g., see in [16] our Hessian-matrix analysis for indefiniteness in MLP-learning and references therein). Since the Hessian matrix of J in Equation (3) can be evaluated efficiently by our recently-developed second-order stagewise backpropagation [13], [15], [18], it would be of our great interest to investigate plateaus in CANFIS neuro-fuzzy modular-network learning.

ACKNOWLEDGMENT

Eiji Mizutani would like to thank Stuart Dreyfus (UC Berkeley) for a countless number of useful discussions on discrete-time optimal-control methods that led to the development of hidden-node teaching.

REFERENCES

- [1] A. Asuncion and D.J. Newman. UCI Machine Learning Repository: <http://www.ics.uci.edu/~mllearn/MLRepository.html>, University of California, Irvine, School of Information and Computer Sciences, 2007.
- [2] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, 2001 (Corrected printing 2002).
- [3] S. Horikawa, T. Furuhashi and Y. Uchikawa. On Fuzzy Modeling Using Fuzzy Neural Networks with the Back-Propagation Algorithm. *IEEE Trans. on Neural Networks*, Vol.3, No.5, pp. 801–806, 1992.
- [4] E. Mizutani, J.-S. R. Jang, K. Nishio, H. Takagi, and D. M. Auslander. Coactive neural networks with adjustable fuzzy membership functions and their applications. In *Proc. of the International Conference on Fuzzy Logic and Neural Networks*, pp. 581–582, Iizuka, Japan, August 1994.
- [5] Eiji Mizutani. Chapter 13: Coactive Neuro-Fuzzy Modeling: Toward Generalized ANFIS. In *Neuro-Fuzzy and Soft Computing: a computational approach to learning and machine intelligence* (J.-S. R. Jang, C.-T. Sun and E. Mizutani), pages 369–400. Prentice Hall, 1997.
- [6] E. Mizutani, S. Dreyfus, and K. Nishio. On derivation of MLP backpropagation from the Kelley-Bryson optimal-control gradient formula and its application. In *Proc. of the IEEE International Conference on Neural Networks (vol.2)*, pages 167–172, Como, Italy, July 2000.
- [7] E. Mizutani and J. W. Demmel. On separable nonlinear least squares algorithms for neuro-fuzzy modular network learning. In *Proceedings of the International Joint Conference on Neural Networks*, pages 2399–2404, Honolulu USA, May 2002.
- [8] E. Mizutani and K. Nishio. Fuzzy Mixtures of Complementary Local Experts: Towards Neuro-Fuzzy Modular Networks. In *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2, 1192–1197, Honolulu USA, May 2002.
- [9] E. Mizutani and S. E. Dreyfus. MLP's hidden-node saturations and insensitivity to initial weights in two classification benchmark problems: parity and two-spirals. In *Proceedings of the International Joint Conference on Neural Networks*, pages 2831–2836, Honolulu USA, May 2002.
- [10] E. Mizutani and K. Nishio. Multi-Illuminant Color Reproduction for Electronic Cameras Via CANFIS Neuro-Fuzzy Modular Network Device Characterization. *IEEE Transactions on Neural Networks*, Vol. 13, No. 4, pages 1009–1022, July 2002.
- [11] E. Mizutani and J. Demmel. On structure-exploiting trust-region regularized nonlinear least squares algorithms for neural-network learning. *Neural Networks*, 16(5-6): 745–753, 2003.
- [12] E. Mizutani and J. W. Demmel. Iterative scaled trust-region learning in Krylov subspaces via Pearlmuter's implicit sparse Hessian-vector multiply. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16 (NIPS 2003)*, volume 16, pages 209–216. MIT Press, 2004.
- [13] E. Mizutani and S. E. Dreyfus. On derivation of stagewise second-order backpropagation by invariant imbedding for multi-stage neural-network learning. In *Proceedings of the IEEE-INNS International Joint Conf. on Neural Networks*, pages 4762–4769, Vancouver, Canada, July 2006.
- [14] E. Mizutani and J.-Y. C. Fan. On exploiting symmetry for multilayer perceptron learning. In *International Joint Conference on neural networks*, pp. 2857–2862, Aug 2007.
- [15] E. Mizutani and S. E. Dreyfus. Second-order stagewise backpropagation for Hessian-matrix analyses and investigation of negative curvature. *Neural Networks*, 2008 Special Issue: Advances in Neural Networks Research: IJCNN 2007, pages 193–203, Vol.21, 2008.
- [16] E. Mizutani and S. E. Dreyfus. An analysis on negative curvature induced by singularity in multi-layer neural-network learning. In *Advances in Neural Information Processing Systems*, Vol. 23, MIT Press, NIPS 2010.
- [17] E. Mizutani and J. W. Demmel. Trust-region variable projection algorithms for separable nonlinear least squares learning. To appear in *Proceedings of the IEEE-INNS International Joint Conference on Neural Networks*, San Jose, California, USA, August 2011.
- [18] Eiji Mizutani. On exploiting sparsity in CANFIS neuro-fuzzy modular network learning by second-order stagewise backpropagation. To appear in *Proceedings of the eighth International Conference on Fuzzy Systems and Knowledge Discovery*, Shanghai, China, July 2011.
- [19] H. Schwenk and Y. Bengio. Boosting Neural Networks. *Neural Computation*. 12: page1869–1887.2000
- [20] E. D. Sontag and H. J. Sussmann. Backpropagation separates when perceptrons do. In *Proceedings of the International Joint Conf. on Neural Networks (IJCNN'89)*, pages 639–642, vol.1, Washington, DC, USA, June, 1989.
- [21] M.Sugeno and G.Kang. Structure Identification of Fuzzy Model. *Fuzzy Sets and Systems*, pp.329–346, Vol.28, 1986.
- [22] T. Takagi and M. Sugeno. Fuzzy identification of systems and its application to modeling and control. *IEEE Trans. on Systems Man and Cybernetics*, Vol.15, pp. 116–132, 1985.