Fuzzy Associative Conjuncted Maps Network

Hanlin Goh, Joo-Hwee Lim, Chai Quek

Fuzzy Associative Conjuncted Maps Network

Hanlin Goh, Joo-Hwee Lim, *Member, IEEE*, and Chai Quek, *Member, IEEE*

***Abstract*—The fuzzy associative conjuncted maps (FASCOM) is a fuzzy neural network that associates data of nonlinearly related inputs and outputs. In the network, each input or output dimension is represented by a feature map that is partitioned into fuzzy or crisp sets. These fuzzy sets are then conjuncted to form antecedents and consequences, which are subsequently associated to form IF–THEN rules. The associative memory is encoded through an offline batch mode learning process consisting of three consecutive phases. The initial unsupervised membership function initialization phase takes inspiration from the organization of sensory maps in our brains by allocating membership functions based on uniform information density. Next, supervised Hebbian learning encodes synaptic weights between input and output nodes. Finally, a supervised error reduction phase fine-tunes the network, which allows for the discovery of the varying levels of influence of each input dimension across an output feature space in the encoded memory. In the series of experiments, we show that each phase in the learning process contributes significantly to the final accuracy of prediction. Further experiments using both toy problems and real-world data demonstrate significant superiority in terms of accuracy of nonlinear estimation when benchmarked against other prominent architectures and exhibit the network's suitability to perform analysis and prediction on real-world applications, such as traffic density prediction as shown in this paper.**

***Index Terms*—Fuzzy associative conjuncted maps (FASCOM), fuzzy associative memory, fuzzy neural networks, Hebbian learning, Iris plant classification, multivariate data analysis, Nakanishi's nonlinear estimation tasks, neurofuzzy systems, supervised learning, traffic density prediction, two-spiral classification, unsupervised learning.**

I. Introduction

FUZZY neural networks [1]–[4] are hybrid intelligent systems that combine the humanlike knowledge representation and reasoning of fuzzy logic [5], with the adaptation and learning capabilities of neural networks. As such, they maximize the desirable properties of both fuzzy logic and neural networks, and become powerful tools for learning and inference. The methods to encode knowledge into fuzzy systems are common difficulties. These methods include membership function determination, fuzzy rule identification, and fuzzy rule inference. These problems may be alleviated by using brain-inspired features, such as self-organization. On the other hand, neural networks are typically black boxes and analysis of a trained network is difficult. Thus, the integration of fuzzy logic in neural networks can significantly improve the transparency of the network. It is hoped that while data prediction remains highly accurate, some form of analysis of the complex multivariate systems they represent is still possible.

A fuzzy neural network belongs to one of two paradigms: 1) linguistic fuzzy model (e.g., Mamdani model proposed in [6]) and 2) precise fuzzy model (e.g., Takagi–Sugeno–Kang (TSK) model proposed in [7]–[9]). In the linguistic model given in (1), both the antecedent and consequence are fuzzy sets, whereby for a rule R_k

$$R_k = \text{if } x_1 \text{ is } A_{k,1} \ \ldots \text{and } x_I \text{ is } A_{k,I} \text{ then } y \text{ is } C_k \tag{1}$$

where $\mathbf{X} = [x_1, \ldots, x_I]^T$ and y are the input vector and output value, respectively, $A_{k,i}$ and C_k represent linguistic labels, and I is the number of inputs. In the precise fuzzy model given in (2) and (3), the antecedent is a fuzzy set and the consequence is formed by linear equations [10], [11]

$$R_k = \text{if } x_1 \text{ is } A_{k,1} \ \ldots \text{and } x_I \text{ is } A_{k,I} \text{ then } y_k = f_k(x) \tag{2}$$

$$y = \sum_{k=1}^{K} w_k y_k \tag{3}$$

where K is the number of rules, and w_k is the strength of R_k. Linguistic fuzzy models focus on interpretability and lack in accuracy, while precise fuzzy models focus on accuracy and lack human interpretability [11], [12].

In this paper, we propose the fuzzy associative conjuncted map (FASCOM) architecture, which is an implementation of a linguistic fuzzy model that uses associations to learn relationships between input and output fuzzy sets. Its main objective is to provide a means for highly accurate data prediction, with some capability of interpretability of the multivariate system it models. Its design follows a line of work on fuzzy associative memories, pioneered by [13]. Though aware of its drawback in terms of space complexity, FASCOM retains the original simple concept of using a fuzzy associative matrix to represent input–output relationships. Its focus is to improve data prediction accuracy through the introduction of several new features in both data representation and its encoding process. For example, typical associative networks may result in nonperfect retrieval, due to crosstalk among nonmutually orthogonal input patterns. We proposed to alleviate this problem by conjuncting antecedent and consequence fuzzy sets, as in

$$R_k = \text{if } x_1 \text{ is } A_{k,1} \ \ldots \text{and } x_I \text{ is } A_{k,I} \text{ then } y_1 \text{ is } C_{k,1} \ \ldots \text{and } y_S \text{ is } C_{k,S} \tag{4}$$

Manuscript received October 25, 2007; revised November 22, 2008 and February 26, 2009; accepted April 05, 2009. First published July 24, 2009; current version published August 05, 2009.

H. Goh is with the Centre for Computational Intelligence, School of Computer Engineering, Nanyang Technological University, Singapore 639798, Singapore and also with the Computer Vision and Image Understanding Department, Institute for Infocomm Research, Singapore 119613, Singapore (e-mail: hlgoh@i2r.a-star.edu.sg).

J.-H. Lim is with the Computer Vision and Image Understanding Department, Institute for Infocomm Research, Singapore 119613, Singapore (e-mail: joohwee@i2r.a-star.edu.sg).

C. Quek is with the Centre for Computational Intelligence, School of Computer Engineering, Nanyang Technological University, Singapore 639798, Singapore (e-mail: ashcquek@ntu.edu.sg).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2009.2023213

where $\mathbf{Y} = [y_1, \ldots, y_S]^T$ is the output vector and S is the number of outputs. In doing so, we intrinsically form representations of multiple antecedents and consequences within the network.

By learning data obtained from past experience, fuzzy neural networks can identify fuzzy IF–THEN rules between the antecedents and consequences of a multivariate system. This is achieved by means of either unsupervised, supervised, or a combination of both unsupervised and supervised techniques.

In unsupervised approaches [14]–[17], learning algorithms that are used to identify fuzzy rules are only dependent on the natural statistics of the data and are not concerned with the relationships between the input and the outputs. This is followed by processing involving the application of neural network techniques (e.g., clustering, self-organization) to adjust identified rules. Due to this reliance on training data, nonrepresentative data may lead to ill-defined rules and hence producing an inaccurate system model.

For supervised approaches [18], [19], supervised learning methods (e.g., backpropagation [20]) are used to identify rules by mapping inputs to outputs. As such, it is sensitive to the correlation between input and output data. However, a drawback for this approach is that the semantics of the fuzzy neural network remains opaque, which contradicts the original aim of combining fuzzy logic with neural networks.

FASCOM is encoded through a learning process consisting of three phases. The first phase involves an unsupervised initialization of membership functions that quantizes the input data. The second phase is a supervised Hebbian learning process that encodes synaptic weights between conjuncted fuzzy sets. The third is a supervised error reduction phase that fine-tunes the partially encoded network and completes the learning process. The entire learning process is performed in an offline batch mode manner, similar to [21] and [22]. This is as opposed to online learning methods [23], [24] where the models are incrementally updated with every new training instance, resulting in fuzzy rules that adaptively evolve with the sequence of training samples. On the other hand, if new data is provided to FASCOM, the entire model needs to be retrained.

The proposed architecture also bears similarity to the popular fuzzy min–max neural networks and its variants [25], [26], whereby hyperboxes are placed within the feature space. A learning algorithm then dynamically adjusts the quantity of these hyperboxes or modify their sizes through and expansion and contraction process, such that the overlap between classes is eliminated. However, in many real-world scenarios, data between classes may not be easily separable. As such, a gradual assessment of membership between classes is also essential, so that the classes are not mutually exclusive [27]. This is also what FASCOM aims to achieve. However, these methods differ with FASCOM in their manner of deriving fuzzy regions. While the fuzzy min–max networks use a supervised approach to assign and adjust fuzzy sets, FASCOM uses the distribution of the data to initialize the fuzzy regions in an unsupervised manner. Supervision is only introduced in the later phases of encoding whereby an associative network is encoded.

Fuzzy IF–THEN rules provide a localized linguistic understanding that is dependent on the locations of their membership functions. But little information can be extracted about the influences of input dimension on the output. This can be somewhat achieved using various well-established methods to analyze and reduce dimensions of a multivariate system, such as principle component analysis (PCA) [28]. PCA is commonly used for dimension reduction, but fails to provide a description of the relationships between the input and output dimensions. Other dimension reduction methods have also been used in the fuzzy neural network literature [12], [35], whereby dimension selection is based on the existence of fuzzy rules identified in the dimensions. However, while all these methods provide a general idea of which dimensions are important and which are not, they do not specify to which portions of the output the inputs exert their influences on. Interestingly, besides being able to identify fuzzy rules for data estimation, the encoded FASCOM network can also provide insights on how the influence of an input dimension varies across an output feature space.

The FASCOM architecture introduces the following key features, with their benefits explained this paper.

1) The unsupervised learning phase (Section III-A) that initializes membership functions based on uniform information density results in an improvement of the output precision of nonlinear estimation as compared to one with a proportional distribution of membership functions in the feature space. Since feature maps are organized based on the information distribution of the data in an unsupervised manner, it is a quick and effective approach for initialization as it only involves a mapping between uniformed and nonuniformed scales of the feature maps. There is evidence that this sort of neuron organization is extensively seen in the sensory maps of humans and other organisms [29]–[34].
2) A supervised error reduction learning phase (Section III-C) that fine-tunes the network is introduced to effectively enhance the accuracy of data prediction. It also presents the influences of an input dimension on different portions of an output feature space. Conversely, we can identify the influences of different inputs on a particular portion of an output. It raises the level of understanding, in addition to fuzzy rules, by allowing the user to determine which factors (or dimensions) are the likely causes of particular ranges of the output. This is especially beneficial for nonlinear systems whereby the resultant output might be influenced differently across output space. Such analytical information can be easily extracted from the network because weights relating an input feature with various portions of the output space are learned in this phase. Unlike feature selection methods that discard noninfluential dimensions based on global statistics, our method is able to retain input dimensions that have strong influences on small portions of the output.
3) Input and output conjunctions reduce and possibly remove crosstalk in nonmutually orthogonal input patterns, as demonstrated in Sections V-A and V-B. This enables the representation of multiple antecedents and multiple consequences, as opposed to singular ones. Furthermore, it not only boosts information representation in an associative

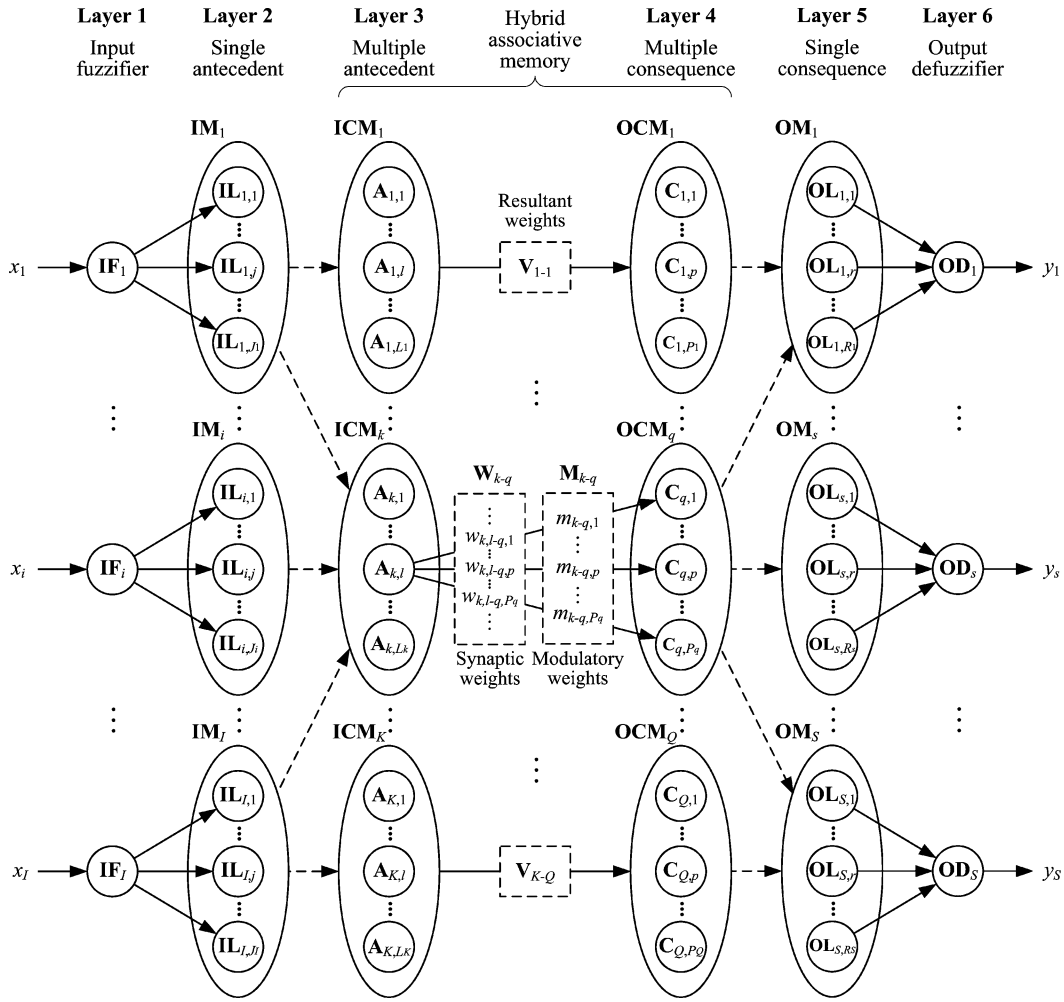


Fig. 1. Structure of the FASCOM network.

neural network, but also enhances the fuzzy-logical linguistic representation. As such, this feature is in essence a true amalgamation of neural and fuzzy concepts.

Through the series of benchmark experiments, we also identified some possible limitations of high complexity in terms of space usage. Like many neural network models, there is also a requirement of parameter setting for optimal results. Based on these factors, we suggest that this architecture is especially suitable for applications that do not need real-time or online learning, but rather require fast and accurate prediction, or those needing descriptive information on relations between input and dimensions for the multivariate system.

II. FASCOM Network Structure

The FASCOM architecture is a six-layer neural network as depicted in Fig. 1. The six layers are, namely, input fuzzifier (layer 1), single antecedent (layer 2), multiple antecedent (layer 3), multiple consequence (layer 4), single consequence (layer 5), and output defuzzifier (layer 6). The outputs of each layer are projected as the inputs of the next layer in a feedforward manner. Layers 1–3 represent input layers, while layers 4–6 represent output layers.

The inputs and outputs of the network are represented as vectors $\mathbf{X} = [x_1, \ldots, x_i, \ldots, x_I]^T$ and $\mathbf{Y} = [y_1, \ldots, y_s, \ldots, y_S]^T$, where I and S denote the number of input and output linguistic variables, respectively. While the output vectors are nonfuzzy, the inputs may be either fuzzy or crisp. Input fuzzification and output defuzzification are automatically accomplished in layers 1 and 6, respectively. Table I lists the description and quantity of the nodes and feature maps in each layer of the network. In biological neural circuitry, feature maps may be subsequently integrated in association areas for purposeful action or cognitive function [36]. Similarly, the inputs of the network are associated with the outputs in the hybrid associative memory between layers 3 and 4. $\mathbf{ICM}$s in layer 3 are linked to $\mathbf{OCM}$s in layer 4 via a $(\mathbf{ICM}_k, \mathbf{OCM}_q)$ association, with each representing a fully connected heteroassociative associative network between $\mathbf{ICM}_k$ and $\mathbf{OCM}_q$. If every $\mathbf{ICM}_k$ is connected to every $\mathbf{OCM}_q$, there will be a total of $(K \cdot Q)$ such associative networks. In each $(\mathbf{ICM}_k, \mathbf{OCM}_q)$ network, there are $(L_k \cdot P_q)$ connections, each representing a fuzzy IF–THEN rule (i.e., if $\mathbf{A}_{k,l}$, then $\mathbf{C}_{q,p}$). Memory between $\mathbf{A}_{k,l}$ and $\mathbf{C}_{q,p}$ is stored as *synaptic weight* $w_{k,l-q,p}$. The synaptic weights

TABLE I
DESCRIPTION AND QUANTITY OF NODES AND FEATURE MAPS IN THE NETWORK

Layer	Node symbol	Node name	Feature map symbol	Feature map name	No. of nodes per feature map	No. of feature maps	Total no. of nodes in network
1	$\mathbf{IF}_i$	Input linguistic node					I
2	$\mathbf{IL}_{i,j}$	Input label node	$\mathbf{IM}_i$	Input map	J_i	I	$\sum_{i=1}^{I} J_i$
3	$\mathbf{A}_{k,l}$	Antecedent node	$\mathbf{ICM}_k$	Input conjuncted map	L_k	K	$\sum_{k=1}^{K} L_k$
4	$\mathbf{C}_{q,p}$	Consequence node	$\mathbf{OCM}_q$	Output conjuncted map	P_q	Q	$\sum_{q=1}^{Q} P_q$
5	$\mathbf{OL}_{s,r}$	Output label node	$\mathbf{OM}_s$	Output map	R_s	S	$\sum_{s=1}^{S} R_s$
6	$\mathbf{OD}_s$	Output linguistic node					S

for a $(\mathbf{ICM}_k, \mathbf{OCM}_q)$ association form a correlation matrix $\mathbf{W}_{k-q}$ defined in

$$\mathbf{W}_{k-q} = \begin{bmatrix} w_{k,1-q,1} & \cdots & w_{k,1-q,p} & \cdots & w_{k,1-q,P_q} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{k,l-q,1} & \cdots & w_{k,l-q,p} & \cdots & w_{k,l-q,P_q} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{k,L_k-q,1} & \cdots & w_{k,L_k-q,p} & \cdots & w_{k,L_k-q,P_q} \end{bmatrix}. \tag{5}$$

Connections are further weighted by a *modulatory weight* $m_{k-q,p}$ that signifies a connection between $\mathbf{ICM}_k$ with $\mathbf{C}_{q,p}$. Modulatory weights between an $(\mathbf{ICM}_k, \mathbf{OCM}_q)$ pair can be structured as a vector given by

$$\mathbf{M}_{k-q} = \left[m_{k-q,1}, \ldots, m_{k-q,p}, \ldots, m_{k-q,P_q}\right]^T. \tag{6}$$

As a result, a connection between nodes $\mathbf{A}_{k,l}$ and $\mathbf{C}_{q,p}$ is weighted by *resultant weight* $v_{k,l-q,p}$, which is the product of $w_{k,l-q,p}$ and $m_{k-q,p}$, and can be computed by

$$v_{k,l-q,p} = m_{k-q,p} \cdot w_{k,l-q,p} \tag{7}$$

or expressed in the following as a correlation matrix $\mathbf{V}_{k-q}$

$$\mathbf{V}_{k-q} = \begin{bmatrix} v_{k,1-q,1} & \cdots & v_{k,1-q,p} & \cdots & v_{k,1-q,P_q} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ v_{k,l-q,1} & \cdots & v_{k,l-q,p} & \cdots & v_{k,l-q,P_q} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ v_{k,L_k-q,1} & \cdots & v_{k,L_k-q,p} & \cdots & v_{k,L_k-q,P_q} \end{bmatrix}. \tag{8}$$

Through the processes of conjunction in layers 3 and 4, and association between the layers, this correlation matrix may end up being large. This problem of redundancy from population coding is also observed in biological systems, with only a small portion of cells in a population coded feature map responding only to stimulation of a specific region. However, it is undesirable in a computational paradigm due to its spatial redundancy. Similarly to [13], since each element within the association matrix represents a fuzzy rule, this also means that there will be a large number of fuzzy rules generated. One way to circumvent this problem is to extract a ranked list of fuzzy rules based on their computed resultant weights, thus enabling the identification and retention of those rules with higher impact and removing noninfluential fuzzy rules. Other methods to tackle this problem have also been introduced in [37]–[40].

Layer 5 has a structure similar to layer 2, and consists of S output maps $\mathbf{OM}_s$. $\mathbf{OM}_s$ has R_s *output label nodes* $\mathbf{OL}_{s,r}, r \in \{1, \ldots, R_s\}$, with each node symbolizing an output linguistic label of an output variable. The total number of nodes in this layer is $\sum_{s=1}^{S} R_s$. Layer 6 consists of S *output linguistic nodes* $\mathbf{OD}_s, s \in \{1, \ldots, S\}$, corresponding to the size of output vector $\mathbf{Y}$, with $\mathbf{OD}_s$ representing the sth output linguistic variable. This layer converts fuzzy outputs from layer 5 to a crisp output.

As a convention, the output of a node or map is denoted as Z with subscripts specifying its origin. For example, $Z_{\mathbf{IF}_i}$ represents the output of node $\mathbf{IF}_i$ and $Z_{\mathbf{IM}_i}$ denotes the output vector of $\mathbf{IM}_i$. All outputs from a layer are propagated to the corresponding inputs at the next layer with unity link-weight λ through an existing connection, except for the connections between layers 3 and 4 where connections are weighted by resultant weight $v_{k,l-q,p}$.

A. Layer 1: Input Fuzzifier

$\mathbf{IF}_i$ nodes in the input fuzzifier layer represent input linguistic variables such as "width," "height," etc. This layer fuzzifies the input vector using the following into a vector of fuzzy membership functions

$$Z_{\mathbf{IF}_i} = \mu_i(x_i) \tag{9}$$

where $\mu_i(\cdot) \in [0, 1]$ is the membership function of $\mathbf{IF}_i$. If a fuzzy input is presented to a node in this layer, the node simply redirects it as the output of the node. This layer first assigns x_i as the centroid of the membership function with value 1. If no smoothing is involved, $Z_{\mathbf{IF}_i} = x_i$. Two smoothing functions were identified for our experiments: *Gaussian* $G(\cdot)$ defined by (10), and *Laplacian of Gaussian* $LoG(\cdot)$ defined by (11)

$$G(i) = \exp\left(-\frac{i^2}{2\sigma^2}\right) \tag{10}$$

$$LoG(i) = \left(1 - \frac{i^2}{2\sigma^2}\right)\exp\left(-\frac{i^2}{2\sigma^2}\right) \tag{11}$$

where σ^2 is the variance of the function. $LoG(\cdot)$ is used to model lateral inhibition in the biological neural circuitry [36], [41],

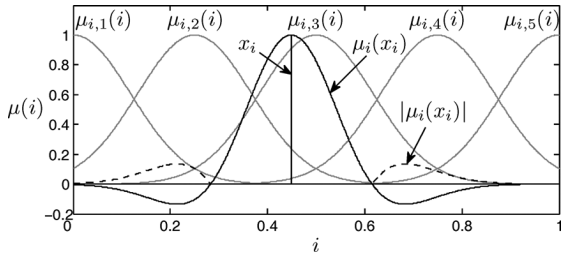


Fig. 2. Computing the fuzzy subsethood measure from a $LoG(\cdot)$ smoothed input x_i.

in which positive signals from the center of a stimuli are surrounded by a ring of negative signals. This is especially useful for increasing signal discrimination by improving the contrast of signals. The *difference of Gaussian* operator is sometimes used as an approximation for $LoG(\cdot)$ [42].

B. Layer 2: Single Antecedent

sIn layer 2, the node $\mathbf{IL}_{i,j}$ represents the jth input label, such as "low," "medium," "high," etc., corresponding to the ith input linguistic variable and grouped into $\mathbf{IM}_i$. This layer receives fuzzified inputs $Z_{\mathbf{IF}_i}$ from layer 1 and computes the *fuzzy subsethood measure*[43], which is the degree of one fuzzy set belonging to another, for each corresponding input label. Using the minimum T-norm operator $T_{\min}$ for the intersection operation, the subsethood measure $Z_{\mathbf{IL}_{i,j}} \in [-1, 1]$ of $\mathbf{IL}_{i,j}$ and $Z_{\mathbf{IF}_i}$ can be approximated using

$$Z_{\mathbf{IL}_{i,j}} = \frac{\max_{x\in i}(\min(Z^{+}_{\mathbf{IF}_i}(x), \mu_{i,j}(x))) - \max_{x\in i}(\min(|Z^{-}_{\mathbf{IF}_i}(x)|, \mu_{i,j}(x))}{\max_{x\in i}(\mu_{i,j}(x))} \tag{12}$$

where $Z^{+}_{\mathbf{IF}_i} \in [0, 1]$ and $Z^{-}_{\mathbf{IF}_i} \in [-1, 0]$ are positive and negative components of $Z_{\mathbf{IF}_i}$, respectively, and $\mu_{i,j}(\cdot)$ is the membership function of input label $\mathbf{IL}_{i,j}$. The output of $\mathbf{IM}_i$ is described by

$$Z_{\mathbf{IM}_i} = \left[Z_{\mathbf{IL}_{i,1}}, \ldots, Z_{\mathbf{IL}_{i,j}}, \ldots, Z_{\mathbf{IL}_{i,J_i}}\right]^T. \tag{13}$$

An example of this computation is illustrated in Fig. 2. Based on the input $x_i = 0.449$ with $\sigma = 0.12$, $\mu_i(x_i)$ is first computed through a $LoG(\cdot)$ smoothing of x_i. A positive vector $\mathbf{SM}^+$ is derived from the subsethood measure computed separately from all five neurons with membership functions $\mu_{i,1}(i)$, $\mu_{i,2}(i)$, $\mu_{i,3}(i)$, $\mu_{i,4}(i)$, and $\mu_{i,5}(i)$, with only positive values of $\mu_i(x_i)$ [i.e., $Z^{+}_{\mathbf{IF}_i}(x)$], while treating the negative values $Z^{-}_{\mathbf{IF}_i}(x)$ as 0. For example, this operation results in $\mathbf{SM}^+ = [0.045, 0.600, 0.968, 0.297, 0.005]^T$. Similarly, a negative vector $\mathbf{SM}^-$ is computed as $[0.135, 0.135, 0.135, 0.135, 0.092]^T$ by ignoring positive membership values and computing the subsethood measure of the absolute value of the negative ones $|\mu_i(x_i)|$. In practical implementation, these two vectors may be computed sequentially and stored separately. Finally, $\mathbf{SM}^-$ is subtracted from $\mathbf{SM}^+$.

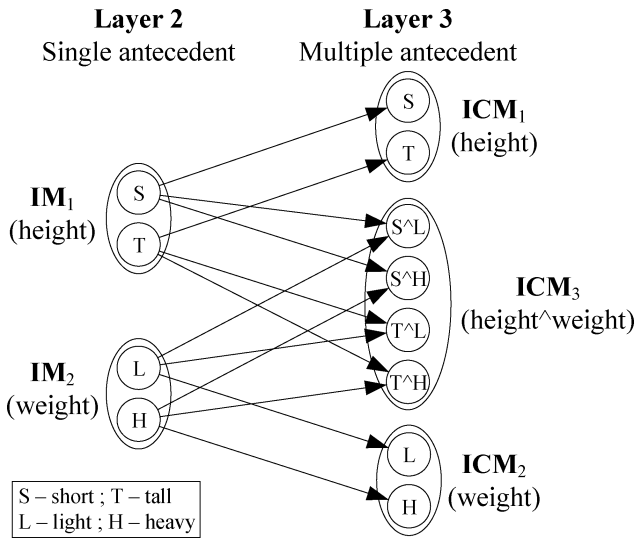


Fig. 3. Example of projecting layer 2 to layer 3.

This essentially combines the two vectors of fuzzy subsethood measures into a single vector as described in (13). As such, the computed $Z_{\mathbf{IM}_i} = [-0.090, 0.465, 0.833, 0.162, -0.087]^T$.

C. Layer 3: Multiple Antecedent

In the multiple antecedent layer, each node $\mathbf{A}_{k,l}$ represents either a single antecedent, such as "if height is short," "if speed is fast," etc., or multiple antecedents, such as "if height is short and weight is light," "if speed is fast and fuel is low and temperature is high," etc. In the example shown in Fig. 3, $\mathbf{ICM}_1$ and $\mathbf{ICM}_2$ representing single antecedents are copies of $\mathbf{IM}_1$ and $\mathbf{IM}_2$. In addition, $\mathbf{ICM}_3$ representing a conjunction of labels of the linguistic variables "height" and "weight" contains four nodes that represent the four multiple antecedents, namely, "if height is short and weight is light," "if height is short and weight is heavy," "if height is tall and weight is light," and "if height is short and weight is heavy."

If $\mathbf{IM}$s are exhaustively conjuncted to form $\mathbf{ICM}$s, the number of $\mathbf{ICM}$s generated will be $\sum_{r=1}^{n}\binom{n}{r}$, where n is the number of input dimensions. This means that the number of $\mathbf{ICM}$s grows exponentially with the number of $\mathbf{IM}$s; as a result, the number of $\mathbf{ICM}$s generated by exhaustive conjunction of all $\mathbf{IM}$s will be significantly larger with every subsequent dimension introduced. The space complexity for the number of $\mathbf{ICM}$ is $O(2^n)$, which poses a limitation on high-dimensional inputs.

To handle data sets with high dimensionality, instead of exhaustively conjuncting all combinations of $\mathbf{IM}$s, the $\mathbf{IM}$s are selectively conjuncted into $\mathbf{ICM}$s. One simple way to do this is to limit the number of preceding $\mathbf{IM}$s that can be combined into each $\mathbf{ICM}$. If the threshold for the number of $\mathbf{IM}$s per $\mathbf{ICM}$ is s, the number of $\mathbf{ICM}$s generated will be $\sum_{r=1}^{s}\binom{n}{r}$. For example, in a data set with ten input dimensions (i.e., ten $\mathbf{IM}$s), the exhaustive conjunction method will produce 1023 $\mathbf{ICM}$s. However, if the number of preceding $\mathbf{IM}$s is limited to two, then the number of $\mathbf{ICM}$s will be 55. This step was introduced in the experiment described in Section V-D3.

The output of each node $Z_{\mathbf{A}_{k,l}} \in [-1, 1]$ is known as the *firing strength*, and results from conjunctions of $\mathbf{IL}_{i,j}$. Using the algebraic product T-norm operator, $Z_{\mathbf{A}_{k,l}}$ is given by

$$Z_{\mathbf{A}_{k,l}} = \prod_{i=1}^{I} \prod_{\substack{j=1 \\ \lambda_{\mathbf{IL}_{i,j}-\mathbf{A}_{k,l}}=1}}^{J_i} Z_{\mathbf{IL}_{i,j}} \tag{14}$$

where $\lambda_{\mathbf{IL}_{i,j}-\mathbf{A}_{k,l}}$ takes on a link weight factor of 1 if there is a connection between $\mathbf{IL}_{i,j}$ and $\mathbf{A}_{k,l}$, and is otherwise 0. Equation (15) describes the output of $\mathbf{ICM}_k$ as a vector of firing strengths

$$Z_{\mathbf{ICM}_k} = \left[Z_{\mathbf{A}_{k,1}}, \ldots, Z_{\mathbf{A}_{k,l}}, \ldots Z_{\mathbf{A}_{k,L_k}} \right]^T. \tag{15}$$

D. Layer 4: Multiple Consequence

A $\mathbf{C}_{q,p}$ node in layer 4 may represent either a single consequence, for example, "then age is young," "then driving is very dangerous," etc., or multiple consequences, such as "then age is young and diet is healthy," "then driving is very dangerous and apply the brake a little," etc. Multiple consequences may be especially useful in applications requiring multiple simultaneous outputs, or data associations with a one-to-many or many-to-many relationship.

Because these nodes represent the "then" clause of a fuzzy IF–THEN rule, when $\mathbf{C}_{q,p}$ is linked with $\mathbf{A}_{k,l}$, a fuzzy IF–THEN rule is formed between them (i.e., if $\mathbf{A}_{k,l}$, then $\mathbf{C}_{q,p}$). This rule can be abbreviated as $\mathbf{A}_{k,l} \rightarrow \mathbf{C}_{q,p}$. Rules are weighted by the resultant weights $v_{k,l-q,p}$, which are determined through the learning processes (details will be described later in Sections III-B and III-C), with rules that have higher weights being more significant than those with lower weights. During recall, the output $Z_{\mathbf{C}_{q,p}} \in [-1, 1]$ of a node $\mathbf{C}_{q,p}$ is its activation level $\phi_{\mathbf{C}_{q,p}}$, as shown in (16), and obtained via the process to be described in Section IV

$$Z_{\mathbf{C}_{q,p}} = \phi_{\mathbf{C}_{q,p}}. \tag{16}$$

E. Layer 5: Single Consequence

In layer 5, $\mathbf{OL}_{s,r}$ represents output labels such as "light," "medium," "heavy" of the corresponding output linguistic variable s. Each node is represented by a membership function $\mu_{s,r}(\cdot)$ similar to the $\mathbf{IL}_{i,j}$ nodes in layer 2. Equation (17) defines a node's output $Z_{\mathbf{OL}_{s,r}} \in [-1, 1]$ as the maximum activation level of consequence nodes $\mathbf{C}_{q,p}$ connected to it

$$Z_{\mathbf{OL}_{s,r}} = \max_{x \in q, y \in p} (\lambda_{\mathbf{C}_{x,y}-\mathbf{OL}_{s,r}} \cdot Z_{\mathbf{C}_{x,y}}) \tag{17}$$

where $\lambda_{\mathbf{C}_{q,p}-\mathbf{OL}_{s,r}}$ is a link weight factor of 1 when there exists a connection between nodes $\mathbf{C}_{q,p}$ and $\mathbf{OL}_{s,r}$, and is 0 otherwise.

F. Layer 6: Output Defuzzifier

Layer 6 consists of $\mathbf{OD}_s$ nodes that represent output variables such as "speed," "temperature," etc. The function of this layer is to convert the fuzzy set outputs of layer 5 into crisp outputs. Three defuzzification schemes are predominant in the literature [44], namely, the center of area, center of maxima, and mean of maxima. A suitable defuzzification scheme s_{DEF}, described in (18), is applied for every $\mathbf{OD}_s$

$$Z_{\mathbf{OD}_s} = s_{\mathrm{DEF}} \left(\sum_{r=1}^{R_s} Z_{\mathbf{OL}_{s,r}} \right). \tag{18}$$

In the subsequent set of experiments, the center of area defuzzification scheme s_{COA} defined in (19) was used

$$s_{\mathrm{COA}} = \frac{\int_S \mu_a(s) s \, dS}{\int_S \mu_a(s) \, dS} \tag{19}$$

where $\int_S \mu_a(s) = \sum_{r=1}^{R_s} Z_{\mathbf{OL}_{s,r}}$ is the aggregated output membership function.

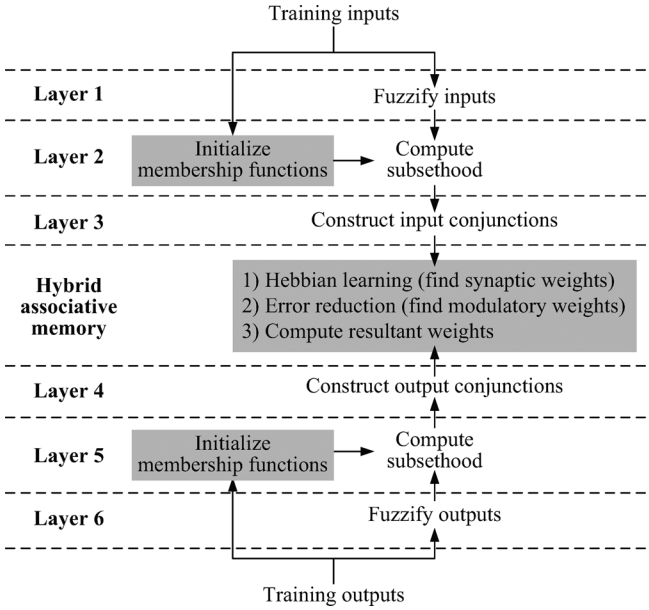


Fig. 4. Learning process (highlighted in gray).

III. Three-Phase Learning Process

Similarly to [45], FASCOM's three-phase learning process consists of three phases. The first phase initializes membership functions for all input and output labels based on uniform information density. Next, Hebbian learning computes the synaptic weights of connections between nodes in layer 3 and those in layer 4. The connections between the input and output conjuncted maps are fine-tuned through a supervised error reduction final phase. As illustrated in Fig. 4, the output layers 4–6 assume the same functional processes as the input layers 1–3, of fuzzification (Section II-A), subsethood measure computation (Section II-B), and conjunction formation (Section II-C), respectively.

To avoid confusion, an alternate symbol $\overleftarrow{Z}$ is used to represent the output of a node or a map that has been derived via a reverse propagation process (i.e., fuzzification for layer 6, subsethood computation for layer 5, and conjunction formation for layer 4). For example, $\overleftarrow{Z}_{\mathbf{OD}_s}$ represents the fuzzified output of $\mathbf{OD}_s$ and $\overleftarrow{Z}_{\mathbf{OCM}_s}$ symbolizes the vector of firing rate of nodes in $\mathbf{OCM}_s$.

A. Unsupervised Membership Function Initialization Phase

A linguistic label is represented by a neuron. In the experimentations, three types of membership functions for these neurons were identified, namely, discrete, rectangular, and Gaussian. The first two offer a crisp representation of feature space, while the third provides a fuzzy representation. We adopt a three-stage approach for membership function initialization: 1) calculation of centroids, 2) initial allocation of neurons, and 3) equalization for uniform information density.

Cyclic and scalar maps are used to represent the two different natures of fuzzy linguistic variables. Examples of cyclic variables include the visual submodalities of hue and orientation. Swindale [46] suggested that cyclic variables may have an advantage that the maps representing these variables can wrap around without compromising continuity and edge effects. For scalar maps containing Gaussian fuzzified neurons, it is necessary to remove edge effects by padding the two ends of the represented space.

1) Computing Centroids of Membership Functions: The number of neurons J in an input or output map representing labels of a linguistic variable i is first fixed. The lower the separability between data classes is, the higher requirement for acuity will be, thus increasing the i required to model the data. A brief discussion on this is presented in Section V-B.

For a neuron j with centroid c_j, its membership function is determined by

$$c_j = \begin{cases} \dfrac{j \cdot (i_{\max} - i_{\min})}{J-1} + i_{\min}, & \text{for scalar variables} \\ \dfrac{j \cdot (i_{\max} - i_{\min})}{J} + i_{\min}, & \text{for cyclic variables} \end{cases} \tag{20}$$

where $i_{\max}$ and $i_{\min}$ are, respectively, the maximum and minimum values of i represented by the map.

2) Initial Uniform Neuron Allocation: The rectangular function for a neuron j defined in (21) results in a crisp representation of information, identical to that of the classical set theory

$$rect_j(i) = \begin{cases} 1, & \text{for } |i - c_j| \le d \\ 0, & \text{otherwise} \end{cases} \tag{21}$$

where c_j is the center of the membership function for neuron j in linguistic variable i, and d defines the distance of the rectangular membership function from its center. The discrete membership function is a special case of rectangular membership functions, where membership functions do not overlap. This occurs when $c_j - c_{j-1} \le 2d$.

Unlike crisp neurons, fuzzy neurons allow for a gradual assessment of membership functions and are biologically similar to the overlapping nature of receptor fields on the periphery receptor sheets. In particular, the Gaussian function, defined in (22), fits the sensitivity curve of the receptive field of a retinal ganglion cell [47]

$$\text{Gaussian}_j(i) = \exp\left[-\frac{(i - c_j)^2}{2\sigma^2}\right] \tag{22}$$

where σ^2 is the variance of the Gaussian function.

3) Uniform Information Density Equalization: There is a disproportion in neuron allocation [36] within various sensory maps in the brain that depends on the usage or sensitivity requirements of a species or an individual. For example, in the auditory cortex of a bat, the region corresponding to the frequency of its biosonar is especially large [29]. This is also observed in other sensory systems, such as the visual [30], [31] and somatosensory [32] systems, and adapted through evolution and individual experiences [33]. It was suggested in [34] that the disproportion is optimized by uniform cortical information density across the map for efficient information transmission. In FASCOM, membership functions are allocated based on the varying information density of the training data across feature space. This disproportion in allocation [Fig. 5(d)] can be achieved by the following four steps:

1) histogram construction based on data points of training data [Fig. 5(a)];
2) Gaussian smoothing of histogram [Fig. 5(a)];
3) histogram equalization to modify the scale of the domain [Fig. 5(b)];
4) using the equalized scale in step 3 to map to the initial neuron allocation [Fig. 5(c)].

With more neurons allocated to the areas with higher information concentration, the output will be more responsive and precise. A similar concept to handle the sensitivity requirements of a fuzzy control system was proposed in [48] to allow for a finer fuzzy control of a truck reversing into a dock, when the truck is nearer to the dock.

B. Supervised Hebbian Learning Phase

1) Fundamental Hebbian Rule: Modified during learning by adopting the Hebbian rule [49], the synaptic weight w_{i-j} stores memory between two neurons i and j. Assuming i and j are nodes in an $\mathbf{ICM}$ and $\mathbf{OCM}$, respectively, the change in synaptic weight Δw_{i-j} can be computed using (23) based on their firing strengths Z_i and $\overleftarrow{\overline{Z}}_j$

$$\Delta w_{i-j} = \begin{cases} 0, & \text{if } (Z_i \le 0 \text{ and } \overleftarrow{\overline{Z}}_j \le 0) \\ |\Delta w_{i-j}|_{\max} \cdot Z_i \overleftarrow{\overline{Z}}_j, & \text{otherwise} \end{cases} \tag{23}$$

where $|\Delta w_{i-j}|_{\max}$ which is the maximum possible absolute change in synaptic weight defined in (25). Equation (24) then updates Δw_{i-j} by aggregating it with the current w_{i-j}

$$w_{i-j} = w_{i-j} + \Delta w_{i-j}. \tag{24}$$

2) Modeling Learning Mechanisms: $|\Delta w_{i-j}|_{\max}$ is determined by the learning mechanisms of synaptic plasticity, forgetting, and long-term facilitation [50], from which the following three concepts governing the behavior of learning were derived.

1) The effects of synaptic plasticity, forgetting, and their resultant combination are approximated as constants through time.
2) The effect of long-term facilitation is initially high, but decays as memory strengthens. When memory stabilizes, the effect of facilitation is insignificant.
3) The initial effect of long-term facilitation is very much greater than that of the resultant effect of synaptic plasticity and forgetting.

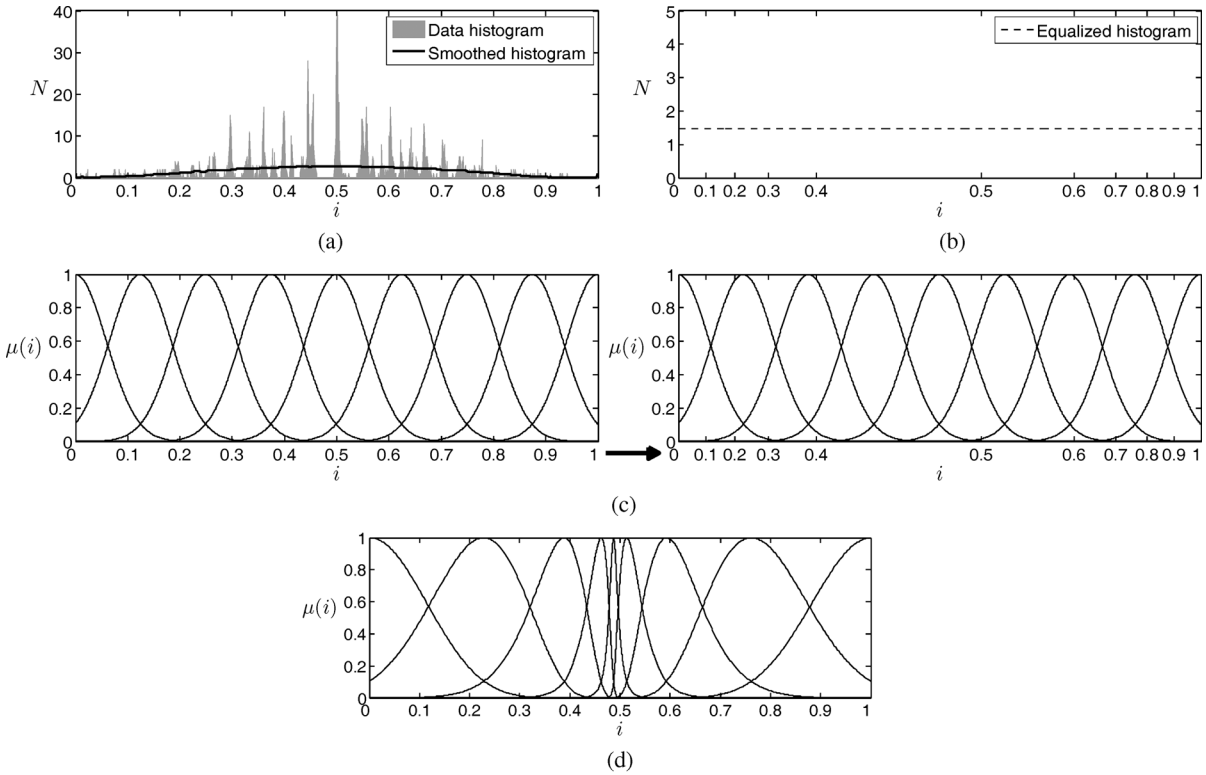


Fig. 5. Steps to achieve membership function allocation based on information density. (a) Steps 1 and 2: insert and smooth data. (b) Step 3: histogram equalization. (c) Step 4: mapping of scale. (d) Final membership function allocation.

The amount of facilitation is dependent on the current strength of memory (i.e., $|w_{i-j}|$), whereby the larger $|w_{i-j}|$ is, the stronger the memory. Since facilitation decreases as memory strengthens, we describe in (25) the effects of the three learning mechanisms as a decaying sigmoid function

$$|\Delta w_{i-j}|_{\max} = \frac{-k_{\mathrm{LTF}}}{1+\exp(-a[|w_{i-j}|-c])} + k_{\mathrm{LTF}} + k_{\mathrm{DC}} \quad (25)$$

where k_{LTF} is the initial effect of facilitation, k_{DC} is the resultant time constant component from the effects of synaptic plasticity and forgetting and a and c define the slope of the sigmoid function. Typically, $k_{\mathrm{LTF}} \gg k_{\mathrm{DC}}$. In cases where the training data is noisy, it is suggested that the value of k_{LTF} be lowered.

Keeping k_{LTF} and k_{DC} constant, the amount of time required for memory development is affected by a and c, such that as c increases or a decreases, the memory will take a longer time to be developed. It is noted however that, if c is set too low or a is set too high, the learning process may be undesirable. Preferably, $c > k_{\mathrm{LTF}} + k_{\mathrm{DC}}$.

When suitable values are selected for the parameters, the three learning mechanisms can be effectively modeled using the sigmoid function shown in Fig. 6. The sigmoid function governs how the maximum possible absolute change in synaptic weight $|\Delta w_{i-j}|_{\max}$ is limited by the current absolute synaptic weight $|w_{i-j}|$ (i.e., strength of memory) between two neurons i and j.

This process of modifying synaptic weights is performed for all training samples, and eventually, depending on the quality of the training data, the learning process will result in some memories being strong like long-term memory and possibly some being weak like short-term memory.

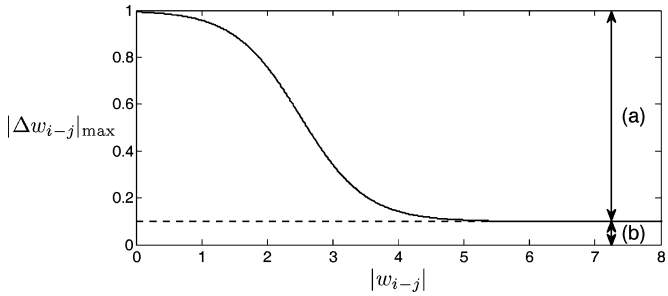


Fig. 6. Effects of learning mechanisms on learning. (a) Effects of long-term facilitation. (b) Combined effects of synaptic plasticity and forgetting.

C. Supervised Error Reduction Phase

After deriving the synaptic weights in the supervised Hebbian learning phase described in Section III-B, the final phase in the learning process reduces output error and fine-tunes the network. This phase discovers the influence of an input conjuncted map $\mathbf{ICM}_k$ on a consequence node $\mathbf{C}_{q,p}$ and stores it as modulatory weight $m_{k-q,p} \in \Re^{\mathrm{nonneg}}$. As shown in Fig. 7, modulatory weights $m_{k-q,p}$ form asymmetrical connections between input and output conjuncted maps such that influence of $\mathbf{ICM}_k$ may differ in magnitude for different portions and output conjuncted map $\mathbf{OCM}_q$. The value of $m_{k-q,p}$ refers to the amount of influence on $\mathbf{C}_{q,p}$ and their meanings are listed in Table II.

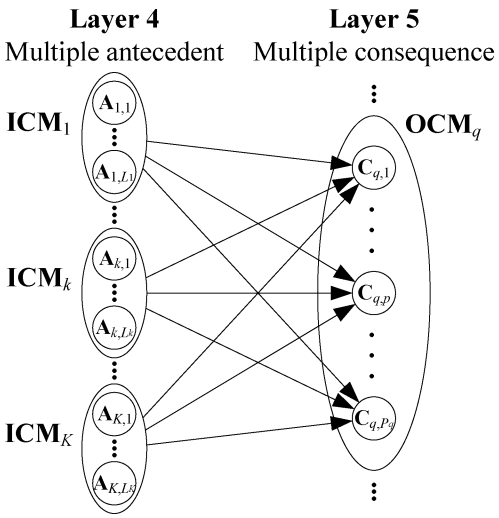


Fig. 7. Modulatory weights form an asymmetrical ($\mathbf{ICM}_k$ to $\mathbf{C}_{q,p}$) relationship between input and output conjuncted maps.

TABLE II
MEANING OF VALUES OF MODULATORY WEIGHTS

Range	Meaning	Action
$m_{k-q,p} = 0$	$\mathbf{ICG}_k$ not required by $\mathbf{C}_{q,p}$	Discarded
$0 < m_{k-q,p} < 1$	$\mathbf{ICG}_k$ is not important to $\mathbf{C}_{q,p}$	Reduced
$m_{k-q,p} = 1$	$\mathbf{ICG}_k$ has an effect on $\mathbf{C}_{q,p}$	Selected
$m_{k-q,p} > 1$	$\mathbf{ICG}_k$ greatly influences $\mathbf{C}_{q,p}$	Enhanced

The flowchart illustrated in Fig. 8 explains the supervised error reduction algorithm, while Table III lists the symbols used in this section.

The algorithm first initializes modulatory weights to 1.0 using (26). For every training input–output pair n, the temporary modulatory weights are assigned by (27). Memory is then recalled using the recalling process explained in Section IV. An initial error for each output node is then computed via (28). Next, using only one input conjuncted map $\mathbf{ICM}_k$ in each iteration, memory is again recalled. A new error for each node is then computed using (29). With this, we apply (30) to derive the change in error with respect to the initial error. Using this information, the temporary modulatory weights for the training instance n are modified using (31). After processing all $\mathbf{ICM}$s for all training samples, the modulatory weights are updated with (32). This entire process is then repeated until a terminating condition is met

$$m_{k-q,p} = 1.0 \qquad \forall k \in K, \forall q \in Q, \forall p \in P_q \tag{26}$$

$$u^{(n)}_{k-q,p} = m_{k-q,p} \qquad \forall k \in K, \forall q \in Q, \forall p \in P_q \tag{27}$$

$$\mathcal{E}^{\text{init}}_{\mathrm{C}_{q,p}} = (\phi_{\mathrm{C}_{q,p}} - \overleftarrow{Z}^{(n)}_{\mathrm{C}_{q,p}})^2 \qquad \forall q \in Q, \forall p \in P_q \tag{28}$$

$$\mathcal{E}^{\text{new}}_{\mathrm{C}_{q,p}} = (\phi_{\mathrm{C}_{q,p}} - \overleftarrow{Z}^{(n)}_{\mathrm{C}_{q,p}})^2 \qquad \forall q \in Q, \forall p \in P_q \tag{29}$$

$$\Delta\mathcal{E}_{\mathrm{C}_{q,p}} = \mathcal{E}^{\text{new}}_{\mathrm{C}_{q,p}} - \mathcal{E}^{\text{init}}_{\mathrm{C}_{q,p}} \qquad \forall q \in Q, \forall p \in P_q \tag{30}$$

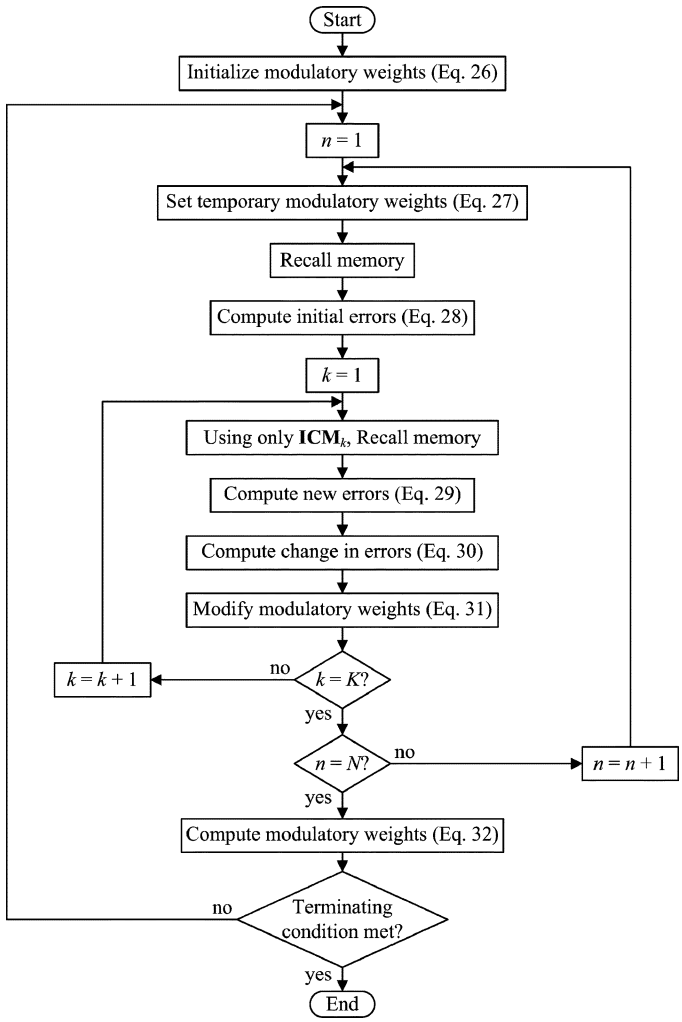


Fig. 8. Supervised error reduction algorithm.

$$u^{(n)}_{k-q,p} = U_{\text{mod}} \cdot u^{(n)}_{k-q,p} \qquad \forall q \in Q, \forall p \in P_q \tag{31}$$

$$m_{k-q,p} = \frac{1}{N}\sum_{n=1}^{N} u^{(n)}_{k-q,p} \qquad \forall k \in K, \forall q \in Q, \forall p \in P_q. \tag{32}$$

The modulatory weights $m_{k-q,p}$ are updated using a modulatory factor U_{mod} defined in (33) based on three factors: 1) the change in squared error, 2) the number of $\mathbf{IM}$ preceding $\mathbf{ICM}_k$, and 3) the current epoch count with respect to the maximum number of epochs. The change in modulation decreases with time and is more sensitive for $\mathbf{ICM}$s with fewer preceding $\mathbf{IM}$. The effects of the change in squared error on the modulatory weights are as follows.

1) Squared-error increases ($\Delta\mathcal{E}_{\mathrm{C}_{q,p}} > 0$) imply that $\mathbf{ICM}_k$ plays an insignificant role, and $u^{(n)}_{k-q,p}$ is reduced.
2) Squared-error decreases ($\Delta\mathcal{E}_{\mathrm{C}_{q,p}} < 0$) imply that $\mathbf{ICM}_k$ influences the output, and $u^{(n)}_{k-q,p}$ is strengthened.
3) Otherwise, $u^{(n)}_{k-q,p}$ is not modified

$$U_{\text{mod}} = \frac{2}{1 + \exp[\delta^{(d \cdot ep)} \cdot \Delta\mathcal{E}_{\mathbf{C}_{q,p}}]} \tag{33}$$

TABLE III
LIST OF SYMBOLS USED IN SUPERVISED ERROR REDUCTION PHASE

Symbol		Description
$m_{k-q,p}$	:	Modulatory weight between $\mathbf{ICM}_k$ and $\mathbf{C}_{q,p}$
$u_{k-q,p}^{(n)}$	:	Temporary value of $m_{k-q,p}$ for the nth training input-output pair
U_{mod}	:	Factor that updates temporary modulatory weights $u_{k-q,p}^{(n)}$
N	:	No. of training input-output pairs
$Z_{\text{L3}} = [Z_{\text{L3}}^{(1)}, \ldots, Z_{\text{L3}}^{(N)}]$	:	Training input vector
$\overleftarrow{Z}_{\text{L4}} = [\overleftarrow{Z}_{\text{L4}}^{(1)}, \ldots, \overleftarrow{Z}_{\text{L4}}^{(N)}]$	:	Training output vector
$Z_{\text{L3}}^{(n)} = [Z_{\mathbf{ICM}_1}^{(n)}, \ldots, Z_{\mathbf{ICM}_K}^{(n)}]$	:	nth training input vector
$\overleftarrow{Z}_{\text{L4}}^{(n)} = [\overleftarrow{Z}_{\mathbf{OCM}_1}^{(n)}, \ldots, \overleftarrow{Z}_{\mathbf{OCM}_Q}^{(n)}]$	:	nth training output vector
$\phi_{\text{L4}} = [\phi_{\mathbf{OCM}_1}, \ldots, \phi_{\mathbf{OCM}_Q}]$	:	Propagated output of layer 4
$\phi_{\mathbf{OCM}_q} = [\phi_{\mathbf{C}_{q,1}}, \ldots, \phi_{\mathbf{C}_{q,P_q}}]$	:	Propagated output of $\mathbf{OCM}_q$
$\phi_{\mathbf{C}_{q,p}}$	:	Activation level of $\mathbf{C}_{q,p}$
$\mathcal{E}_{\mathbf{C}_{q,p}}^{\text{init}}$	:	Initial squared-error for $\mathbf{C}_{q,p}$
$\mathcal{E}_{\mathbf{C}_{q,p}}^{\text{new}}$	:	New squared-error for $\mathbf{C}_{q,p}$
$\Delta\mathcal{E}_{\mathbf{C}_{q,p}}$	:	Squared-error change for $\mathbf{C}_{q,p}$

where $\delta(\delta > 1)$ is a parameter representing the initial average learning rate, d is a variable affected by the number of preceding $\mathbf{IM}$ defined in (34), and ep is the epoch factor defined in (35)

$$d = \begin{cases} 1+\alpha, & \text{if } \alpha \geq 0 \\ \frac{1}{1+|\alpha|}, & \text{otherwise} \end{cases} \tag{34}$$

where $\alpha = (I+1)/2 - \sum_{i=1}^{I}(\lambda_{\mathbf{IM}_i - \mathbf{ICM}_k})$, and $\lambda_{\mathbf{IM}_i - \mathbf{ICM}_k}$ represents a link weight factor of 1 when a connection exists between $\mathbf{IM}_i$ and $\mathbf{ICM}_k$, and 0 otherwise

$$ep = (1 - \frac{\tau}{\tau_{\max}})^2 \tag{35}$$

where τ and $\tau_{\max}$ are the current and maximum number of epochs, respectively.

The two terminating conditions for the algorithm are as follows:

1) maximum number of epochs reached (i.e., $\tau = \tau_{\max}$);
2) when the total change in error reaches zero or falls below a low threshold (i.e., $\sum_{q=1}^{Q}\sum_{p=1}^{P_q}\Delta\mathcal{E}_{\mathbf{C}_{q,p}} \simeq 0$).

IV. RECALLING PROCESS

In the recalling process shown in Fig. 9, an input is first presented to layer 1 of an encoded network. The network will propagate this input to obtain the firing rates for nodes in layer 3 (see Sections II-A–II-C). Memory recall from the hybrid associative memory is then performed and the activation levels for nodes in layer 4 are obtained. Finally, based on output functions, the estimated output is produced at layer 6 (see Sections II-D–II-F). This section focuses on the local processes of each neuron in the memory recall stage.

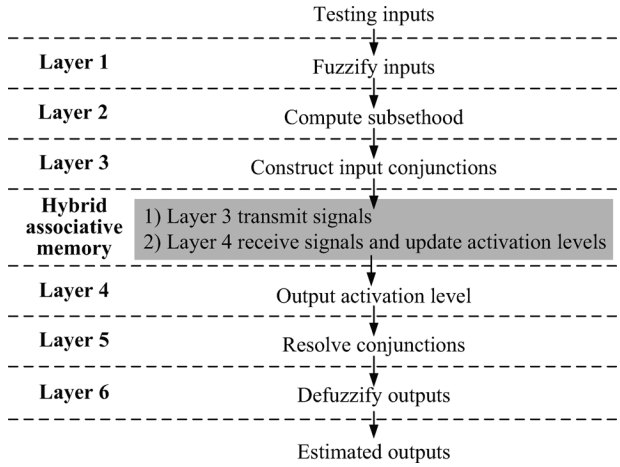


Fig. 9. Recalling process (highlighted in gray).

Assuming neuron i is an $\mathbf{A}_{k,l}$ node in layer 3 and neuron j is a $\mathbf{C}_{q,p}$ node in layer 4. Neuron i transmits a signal based on its firing strength Z_i defined by

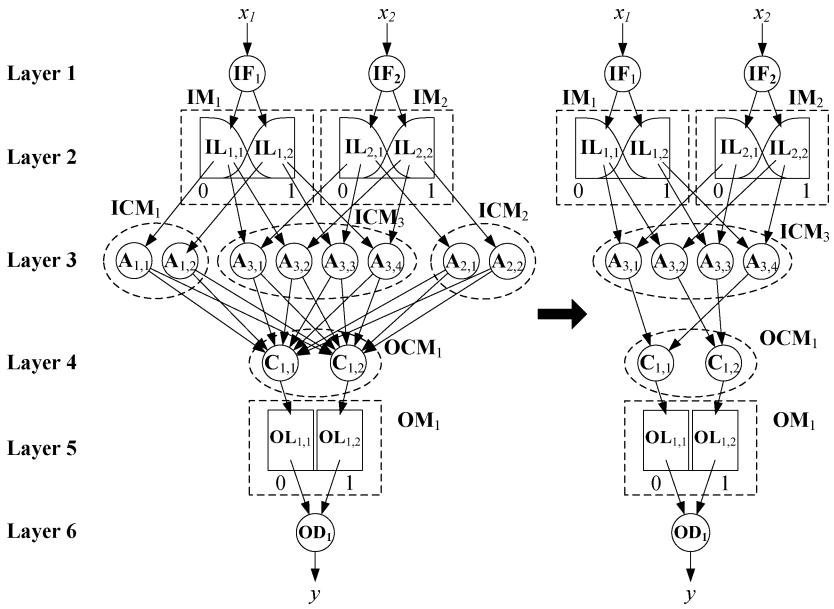
$$a_i = \begin{cases} 0, & \text{for } d_{\min}^{-} \leq Z_i \leq d_{\min}^{+} \\ k_{\text{prop}} \cdot Z_i, & \text{otherwise} \end{cases} \tag{36}$$

where a_i is the output of i, $k_{\text{prop}} \in [0,1]$ is a propagation factor to control network stability, and $d_{\min}^{+} \in [0,1]$ and $d_{\min}^{-} \in [-1,0]$ are predefined positive and negative thresholds.

Neuron j then receives these output signals and updates its activation level ϕ_j based on

$$\phi_j = \sum_{i=1}^{N} v_{i-j} \cdot a_i \tag{37}$$

where v_{i-j} is the resultant weight between i and j, and N is the number of i nodes connected to j.



(a) Initial network (b) Encoded netwotk

Fig. 10. Encoding transformation for the XOR problem.

V. Experiments and Multivariate Analysis

Seven experiments were conducted to perform multivariate analysis and evaluate the performance of nonlinear estimation. The experiments performed were: 1) the exclusive-OR (XOR) problem, 2) the two-spiral problem, 3) iris plant classification, 4) Nakanishi's nonlinear estimation tasks (three experiments), and 5) highway traffic density prediction and analysis.

A. The Exclusive-OR Problem

The exclusive-OR (XOR) problem is a simple nonlinear classification problem with nonmutually orthogonal inputs. This experiment's objective is to observe the behavior of the supervised error reduction phase (Section III-C).

The initial network shown in Fig. 10(a) was designed with two **IM**s, each partitioned using two Gaussian membership functions, and an **OM** with discrete partitioning using two rectangular membership functions. The supervised Hebbian learning phase and the supervised error-reduction phase were then performed. Based on the modulatory weights between the three input conjuncted maps and consequence node $\mathbf{C}_{q,p}$ as shown in Fig. 11, the modulatory weights stabilized at $[0, 0, 1.73]^T$ for $\mathbf{C}_{1,1}$ after six epochs, implying that only $\mathbf{ICM}_3$, representing conjunctions of x_1 and x_2, influenced the output. $\mathbf{ICM}_1$ and $\mathbf{ICM}_2$ do not influence the output. An identical behavior was observed between $\mathbf{C}_{1,2}$ and **ICM**s. After learning, the network discarded $\mathbf{ICM}_1$ and $\mathbf{ICM}_2$.

In the encoded network shown in Fig. 10(b), four connections remain between $\mathbf{ICM}_3$ and $\mathbf{OCM}_1$. These four connections represent four equally weighted fuzzy rules corresponding to the expected behavior of XOR logic as follows:

1) if x_1 is "low" and x_2 is "low," then y is "low";
2) if x_1 is "low" and x_2 is "high," then y is "high";
3) if x_1 is "high" and x_2 is "low," then y is "high";
4) if x_1 is "high" and x_2 is "high," then y is "low."

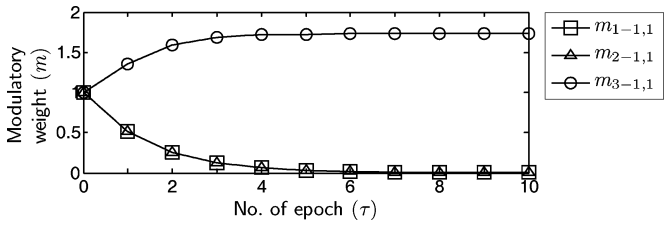


Fig. 11. Changes in modulatory weights between input conjuncted maps and consequence node $\mathbf{C}_{q,p}$.

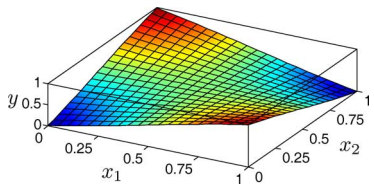


Fig. 12. Output surface showing correct XOR classification.

The XOR problem was correctly classified (Fig. 12) mainly due the existence of conjunctions formed in $\mathbf{ICM}_3$. Without $\mathbf{ICM}_3$, the classification output obtained for all input combinations would be ambiguous (i.e., $y = 0.5$). This shows the importance of layer 3 for removing crosstalk in nonmutually orthogonal input patterns.

B. The Two-Spiral Problem

The classical two-spiral problem [51] is another classical nonlinear classification problem that involves the nonlinear classification of two intertwined spirals, each being labeled as a separate class. As shown in Fig. 13, the training set consists of 194 points (97 per class) and test set has 770 points (385 per class). These sample points are mathematically defined in

$$\underset{(y=1)}{\text{Spiral 1:}} \begin{cases} x_1 = \gamma \sin\theta \\ x_2 = \gamma \cos\theta \end{cases} \quad \underset{(y=0)}{\text{Spiral 2:}} \begin{cases} x_1 = -\gamma \sin\theta \\ x_2 = -\gamma \cos\theta \end{cases} \tag{38}$$

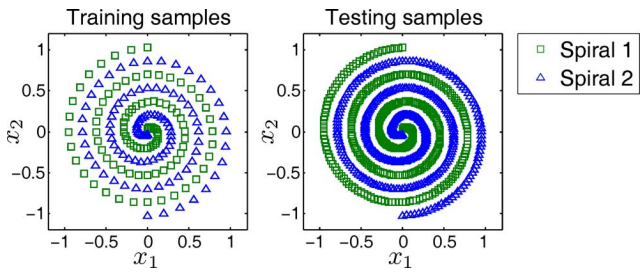


Fig. 13. Two-spiral problem.

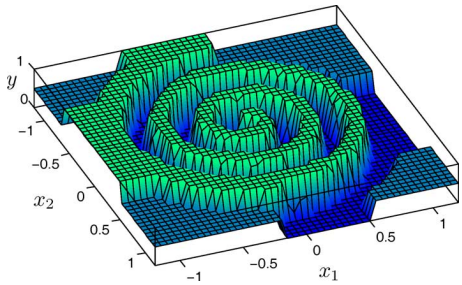


Fig. 14. Output surface depicting correct classification of the two spirals.

where $\gamma = (1/6\pi)(\theta + 2/\pi)$, and $\theta = k\pi/16, k = 0, 1, 2, \ldots, 96$ for the training set or $\theta = k\pi/64, k = 0, 1, 2, \ldots, 384$ for the test set.

A configuration similar to that of the XOR problem (described in Section V-A) was used. However, as opposed to the XOR data, the data for the two-spiral problem requires higher acuity to model due to lower class separability. The number of Gaussian membership functions needed would thus be bigger for this data set.

It was determined empirically that the most suitable numbers of Gaussian membership functions were 12 and 16 for input maps $\mathbf{IM}_1$ and $\mathbf{IM}_2$, which fuzzify the inputs x_1 and x_2, respectively. After encoding, the modulatory weights were discovered to be $[0, 0, 3.4]^T$ for $\mathbf{C}_{1,1}$ and $\mathbf{C}_{1,2}$. Thus, similarly to the XOR experiment, the output was only dependent on the conjunction of inputs. Before the encoding process, there were 660 $(\mathbf{A}_{k,l}, \mathbf{C}_{q,p})$ connections between layers 3 and 4 in the network. After training, 137 positively weighted fuzzy IF–THEN rules were retained. This dramatic increase in the number of rules discovered, as compared to the four discovered for the XOR experiment, was due to the differences in the differences in separability of the data.

FASCOM was able to classify the two spirals across the 2-D space (Fig. 14) and is comparable to the other state-of-the-art architectures benchmarked against in Table IV.

C. Iris Plant Classification

The classification of iris plant uses a popular data set [52] consisting of four input attributes (sepal width and length, and petal width and length) measured off iris plants to determine their subspecies (Iris setosa, Iris versicolor, and Iris virginica). The data set consists of 150 samples, with 50 samples per iris subspecies. More detailed properties and analysis of this data set were recently described in [53].

TABLE IV
BENCHMARKING RESULTS FOR THE TWO-SPIRAL PROBLEM

Architecture	Classication Rate(%)
FASCOM	**100.00**
SVM [21]	100.00
Fuzzy ARTMAP [54]	100.00
GenSoFNN [55]	100.00
FCMAC-CRI [56]	100.00
Direct-output FCMAC-AARS [57]	100.00
FCMAC-Yager [58]	99.87
TSK^0-FCMAC [59]	99.87
Fuzzy-output FCMAC-AARS [57]	99.61
RSPOP [12]	97.27
Lang's 2-5-5-5-1 [51]	92.80

TABLE V
IRIS PLANT CLASSIFICATION RESULTS

Architecture	Cong A	Cong B
FASCOM	**97.33%**	**94.67%**
SVM [21]	96.67%	93.17%
FITSK [60]	96.67%	90.00%
FCMAC-Yager [58]	96.00%	92.00%
FCMAC-CRI [56]	93.66%	93.66%
Fuzzy Output FCMAC-AARS [57]	94.67%	89.83%
TSK-FCMAC [59]	94.67%	92.83%
FALCON [59]	94.67%	92.83%

The FASCOM network was initialized with the four input maps ($\mathbf{IM}_1$ to $\mathbf{IM}_4$) representing sepal width, sepal length, petal width, and petal length, respectively. They contain four, four, eleven, and eleven Gaussian membership functions, respectively. The output map $(\mathbf{OM}_1)$ was partitioned into three discrete sets, with each set representing an iris plant subspecies (setosa, versicolor, and virginica).

In our experiments, we initialized the data set into two different configurations. Configuration A used 100 randomly selected samples for training and the other 50 for testing. Configuration B consisted of 50 randomly selected training samples, while the remaining 100 samples formed the testing set. Six different prediction runs were performed on different randomized instances for the two configurations, with the classification results averaged across all six runs and presented in Table V. As concluded from the results listed in Table V, FASCOM managed to outperform all other benchmarked architectures for both configurations of this classification experiment.

The total number of antecedent-consequent connections $(\mathbf{A}_{k,l} \rightarrow \mathbf{C}_{q,p})$ formed by the network was 5808. Subsequently, after training and pruning off the negatively and zero-weighted connections, the number of resultant fuzzy

TABLE VI
BENCHMARKING RESULTS ON NONLINEAR ESTIMATION FOR THE NAKANISHI ESTIMATION TASKS

Architecture	Example A		Example B		Example C	
	MSE	R	MSE	R	MSE	R
FASCOM	**0.181**	**0.929**	$\mathbf{8.170 \times 10^3}$	**0.999**	**13.930**	**0.953**
Hebb-RR	0.185	0.911	2.423×10^4	0.998	15.138	0.947
SVM	0.258	0.876	2.423×10^5	0.993	29.510	0.925
RSPOP-CRI	0.383	0.856	2.124×10^5	0.983	24.859	0.922
DENFIS	0.411	0.805	5.240×10^4	0.995	69.824	0.810
POP-CRI	0.270	0.877	5.630×10^5	0.946	76.221	0.733
ANFIS	0.286	0.853	2.969×10^6	0.780	38.062	0.875
Mandami	0.862	0.490	6.580×10^5	0.937	40.839	0.865
EFuNN	0.566	0.720	7.247×10^5	0.946	72.541	0.756
Turksen (IVCRI)	0.706	0.609	2.581×10^5	0.993	93.022	0.661
Sugeno (P-G)	0.467	0.845	1.931×10^6	0.990	168.903	0.700

IF–THEN rules obtained was 257.83 when averaged across the six runs.

D. Nakanishi's Nonlinear Estimation Tasks

In the previous two experiments (Sections V-A and V-B), the data contain no noise because our objective was to observe the behavior and outcome of learning. However, noise is present in the next three experiments that use the Nakanishi data set [35]. The data set consists of three examples of real-world nonlinear estimation tasks, namely: 1) a nonlinear system, 2) the human operation of a chemical plant, and 3) the daily stock price data of a stock in a stock market.

The objectives of these experiments is to perform multivariate analysis, comparing the similarity and differences of this analysis with other existing methods, as well as to benchmark data estimation on noisy data against other state-of-the-art architectures. Two measures were used for evaluation of data estimation, namely, the mean squared error (MSE) and the Pearson product-moment correlation coefficient (R). The nonlinear estimation results were evaluated and compared against results obtained from Hebb-RR [61], SVM [21], RSPOP-CRI [12], DENFIS [24], POP-CRI [62], ANFIS [22], [63], Mamdani [6], [64], EFuNN [23], Turksen [65], and Sugeno (P-G)[9]. As shown in Table VI, FASCOM obtained the best results for all three nonlinear estimation tasks.

For the three tasks, the number of fuzzy IF–THEN rules discovered after retaining only the positive $(\mathbf{A}_{k,l}, \mathbf{C}_{q,p})$ connections were 149, 161, and 678, respectively.

1) Task A: A Nonlinear System: The aim of this task is to model a nonlinear system given by

$$y = (1 + x_1^{-2} + x_2^{1.5})^2 : (1 \leq x_1, x_2 \leq 5). \tag{39}$$

Here, four input dimensions are used to predict the value of the 1-D output.

The supervised error reduction phase discovered the influence of individual input conjuncted maps **ICM** on the output y, across output space. As an example, the varying influence of two input conjuncted maps **ICM** representing $x_1 \wedge x_2$ and $x_2 \wedge x_3$ across output space is shown in Fig. 15. It can be observed that $x_1 \wedge x_2$ significantly influenced the output across the entire output feature space, especially for outputs of around 2.0, 2.5, and 4.75. On the other hand, $x_2 \wedge x_4$ only influences the low outputs or below 1.5 and its effects are reduced for the other portions of output space.

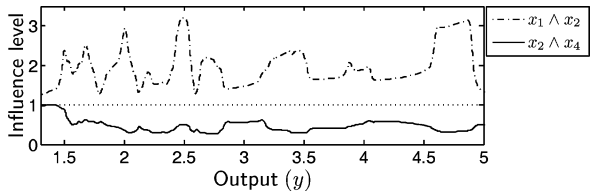


Fig. 15. Two examples showing the varying influence of an input across the output space.

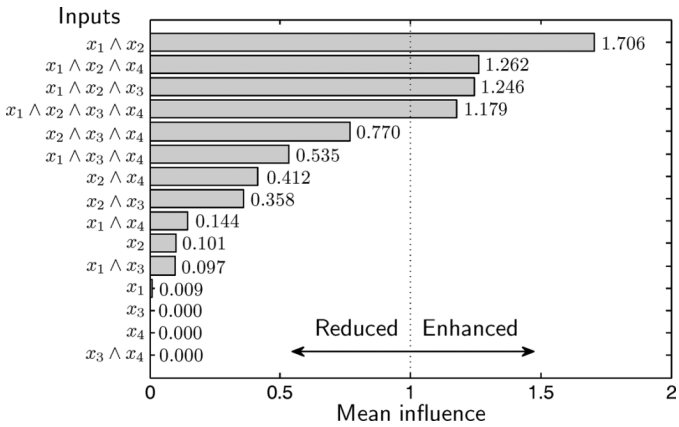


Fig. 16. Mean influence of each input computed across the output space.

For a more generalized description, the mean of the influences is computed across the output feature space and illustrated in Fig. 16. **ICM**s representing $x_1 \wedge x_2$, $x_1 \wedge x_2 \wedge x_4$, $x_1 \wedge x_2 \wedge x_3$, and $x_1 \wedge x_2 \wedge x_3 \wedge x_4$ were generally enhanced across the output space. On the other hand, it is clear that those **ICM**s representing x_3, x_4, and $x_3 \wedge x_4$ exerted no influence on the

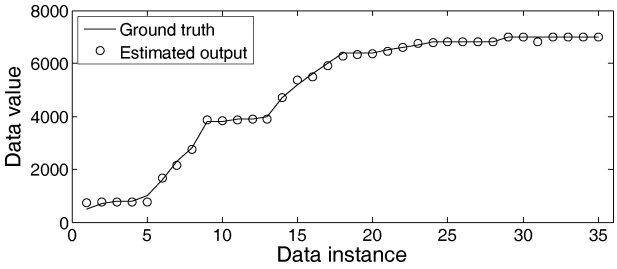


Fig. 17. Output estimation for task B has high precision.

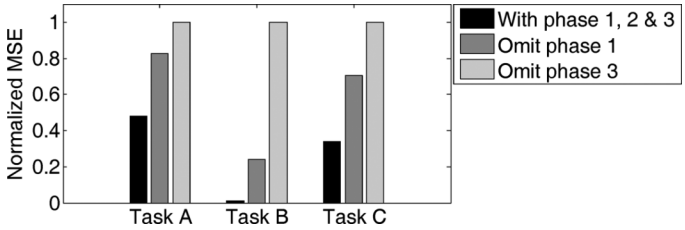


Fig. 18. Comparison between the three different experimental setups for the three Nakanishi estimation tasks.

outcome of y and could be discarded. The effects of all other **ICM**s were reduced.

This generalized outcome is similar to the feature selection method in [35], which discarded the inputs x_3 and x_4, as well as that of [12], which discarded input x_3 and partially discarded input x_4. However, FASCOM was more specific and descriptive in that it was able to provide information of how the influences of inputs and conjunctions of inputs vary across output space. From another perspective, FASCOM identifies the significance of various inputs for a particular portion of the output space.

2) Task B: Human Operation of a Chemical Plant: This task involves the human operation of a chemical plant, whereby there are five input dimensions representing monomer concentration (x_1), charge of monomer concentration (x_2), monomer flow rate (x_3), and local temperatures inside the plant (x_4, x_5), as well as one output dimension representing the set point for monomer flow rate (y).

The supervised error reduction phase identified influences of each input conjuncted map **ICM** across the output space, from which the **ICM**s representing x_2, x_4, x_5, $x_2 \wedge x_4$, $x_2 \wedge x_5$, $x_4 \wedge x_5$, and $x_2 \wedge x_4 \wedge x_5$ were deemed noninfluential to output prediction and were discarded. As a comparison, the feature selection method in [35] discarded inputs x_2, x_4, and x_5, and [12] discarded inputs x_1, x_2, and x_5. The **ICM** that was most enhanced had a modulatory weight of 3.361, and represented the conjuncted input $x_1 \wedge x_3$. From this, we deduce that the set point for monomer flow rate (y) depends mainly on the combination of monomer flow rate (x_3) and monomer concentration (x_1) (i.e., $x_1 \wedge x_3$).

Additionally, as seen from Table VI, FASCOM significantly outperformed all other architectures for this nonlinear estimation task, with a reduction of MSE by 66.3% over the state-of-the-art. The precision of FASCOM's output estimation is displayed in Fig. 17.

3) Task C: Daily Stock Price Data of a Stock in a Stock Market: In this task, ten inputs are used to predict the price of a stock y. The inputs represent the past and present moving averages over a middle period (x_1, x_2), past and present separation ratios with respect to moving average over a middle period (x_3, x_4), present change of moving over a short period (x_5), past and present changes of price (x_6, x_7), past and present separation ratios with respect to moving average over a short period (x_8, x_{10}), and present change of moving average over a long period (x_9).

As mentioned in Section II-C, high-dimensionality data induces high spatial complexity. Exhaustive conjunction on this data set with ten input dimensions would result in 1023 **ICM**s. This increases training time to impractical levels. To handle this problem, a selective approach was performed to reduce the number of **ICM**s generated by limiting the number of preceding **IM**s that would combine to form each **ICM** to two. This results in only 55 ICMs generated and greatly reduces both the spatial resources required and the training time back to practical levels.

For this task, the supervised error reduction phase discovered that inputs corresponding to x_1, x_2, x_3, x_5, x_6, x_8, and x_9 had extremely low influences and their effects were significantly reduced. **ICM**s representing $x_4 \wedge x_{10}$ and $x_4 \wedge x_7$ were discovered to be the most influential ones. This means that the present separation ratio with respect to moving average over a middle period (x_4), present separation ratio with respect to moving average over a short period (x_{10}), and present change of price (x_7) are the main indicators for predicting daily stock prices. As a comparison, Nakanishi *et al.* [35] discarded inputs x_1, x_2, x_3, x_6, x_7, x_9, and x_{10}, and [12] discarded x_1, x_2, x_3, x_6, and x_{10}, and partially discarded x_5, x_7, x_8, and x_9. Interestingly, although there are some differences in the outcome, x_4 was not discarded for all three methods. This means that all three methods considered x_4 to be of importance for output prediction.

We further found that when comparing the past and present values of the same indicators, (i.e., $x_1 - x_2, x_3 - x_4, x_6 - x_7, x_8 - x_{10}$), with the exception of the moving average over a middle period indicator $(x_1 - x_2)$, the present indicators generally have greater impact as compared to the past ones.

The good prediction results for this task (Table VI) suggest that similarly to [66] and [67], FASCOM can be used to perform prediction of the stock market. In addition, because of the natural cause-and-effect interpretation of associations and the descriptive nature of input–output dimension relationships, FASCOM may be suitable for understanding the dynamics of a stock market.

4) Justification of Learning Phases: Besides multivariate analysis and nonlinear estimation, the three Nakanishi nonlinear estimation tasks were also used to analyze the effects of three different experimental setups of the learning process:

1) using all three phases in the learning process;
2) omitting unsupervised membership function initialization (phase 1);
3) omitting supervised error reduction (phase 3).

For each nonlinear estimation task, the resultant MSE produced by each of the three initializations was computed and normalized with respect to the highest MSE produced. As observed from Fig. 18, for all three tasks, the comparison between the

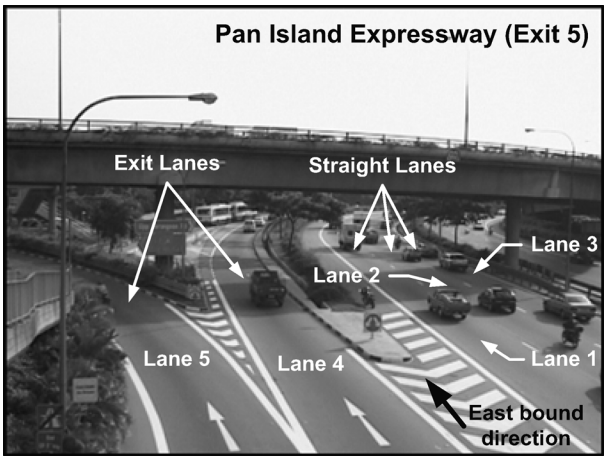


Fig. 19. Photograph of site 29 where the traffic data was collected.

three different experimental setups shows that the inclusion of both phases 1 and 3 produced an MSE that is significantly lower than when either one is omitted. This shows that both phases 1 and 3 are crucial in reducing the MSE of the output estimation and justifies the existence of both phases within the learning process.

E. Highway Traffic Density Prediction and Analysis

The raw traffic data [68] were collected at site 29 located at Exit 5 along the east bound direction of the Pan Island Expressway (PIE) in Singapore, using loop detectors embedded beneath the road surface (Fig. 19). Data spanning a period of six days from September 5 to 10, 1996, for three straight lanes, were considered for this experiment. The purpose of this experiment is to model the traffic flow trend at the site, and subsequently produce predictions of the traffic density of each lane at time $t+\tau$, where $\tau = 5$, 15, 30, 45, 60 min. From this, we can analyze the performance of data prediction and perform multivariate analysis on real-world noisy data.

The data set has four input dimensions, with one representing the normalized time, and three others representing the traffic density of each of the three lanes.

The raw data records the traffic density at 1372 instants with intervals of 5 min between successive instants. After processing the data for forward projections of $\tau = 5$, 15, 30, 45, 60 min, there are 1371, 1369, 1366, 1363, and 1360 paired samples for each set. To evaluate the performance of predicted traffic density, three cross-validation groups (CV1, CV2, and CV3) of training and test sets were formed. Each cross-validation group has 550 training samples each, and 821, 819, 816, 813, and 810 test samples, respectively.

The MSE was computed and averaged across all prediction runs, resulting in "Avg MSE." Also, the "Var" indicator reflecting the consistency of predictions over different time intervals across the three lanes was computed as the change in the mean of Pearson product-moment correlation coefficient from $\tau = 5$ min to $\tau = 60$ min expressed as a percentage of the former. This was then averaged across all three lanes to produce "Avg Var." From the example in Fig. 20, we observe that the prediction accuracy decreases as the time interval τ increases.

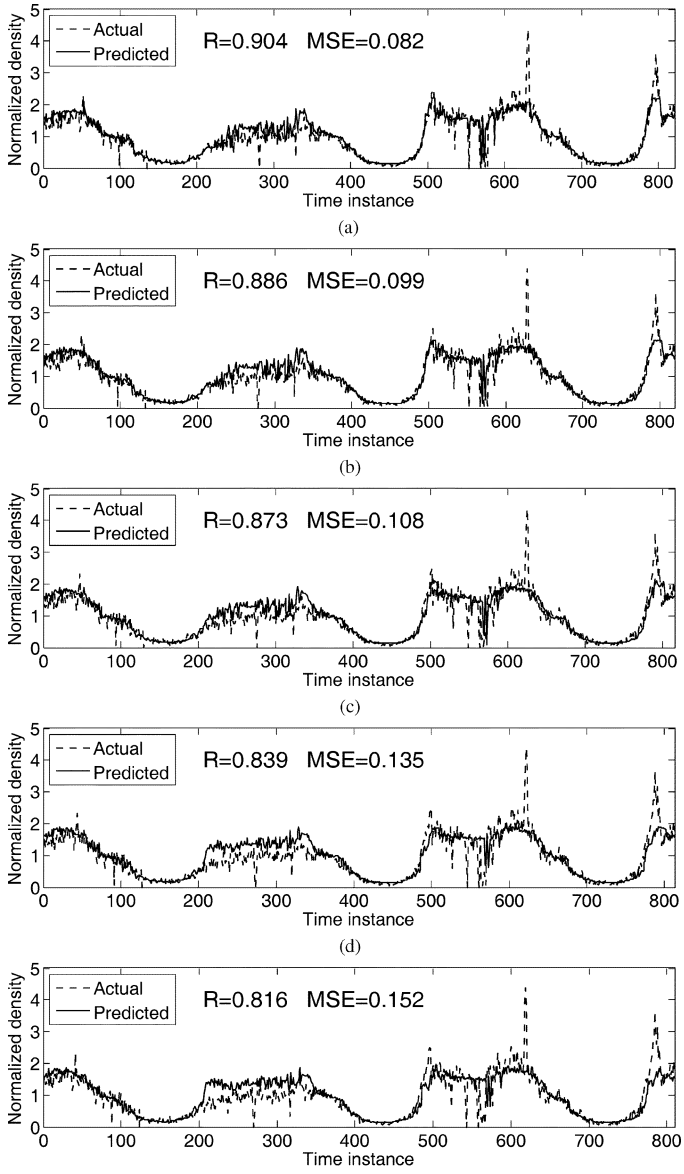


Fig. 20. Traffic density prediction of lane 1 using CV1. (a) Prediction at $\tau = 5$ min. (b) Prediction at $\tau = 15$ min. (c) Prediction at $\tau = 30$ min. (d) Prediction at $\tau = 45$ min. (e) Prediction at $\tau = 60$ min.

The results of traffic density prediction were compared to Hebb-RR [61], SVM [21], RSPOP [12], POP-CRI [62], DENFIS [24], GeSoFNN [55], and EFuNN [23]. FASCOM significantly outperformed all other architectures based on the results shown in Fig. 21, with the lowest "Avg MSE" (0.098) relative to "Avg Var" (19.0%) as compared to other architectures. The results indicate that the output prediction by FASCOM is both highly accurate and consistent over different time intervals. This is desirable.

By analyzing the influence of the input conjuncted maps across the output, it could be deduced that the prediction of traffic densities for a particular lane is mostly dependent on normalized time combined with the current density of that lane, as well as normalized time combined with the density of an adjacent lane. The influence of the density of lanes decreases with distance. Interestingly, the above factors are more distinctive for outputs predicting high traffic density.

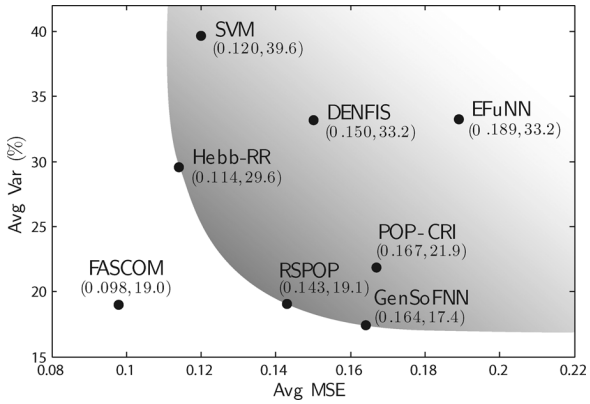


Fig. 21. FASCOM outperforms other benchmarked architectures for highway traffic density prediction.

However, performance comes with a price. On the average, FASCOM extracted 42.3 fuzzy IF–THEN rules from the data. The average number of rules the other architecture extracted was: 8.1 by Hebb-RR, 14.4 by RSPOP, 40.0 by POP-CRI, 9.7 by DENFIS, 50.0 by GeSoFNN, and 234.5 by EFunN. It should also be noted that FASCOM is spatially more complex than most of the other architectures because of the deliberate use of exhaustive conjunctions and associations for information representation and input–output relation. Finally, FASCOM's learning phase can only be performed in an offline batch mode manner, while some other methods, such as EFuNN [23] and DENFIS [24], can perform learning in an online and incremental manner. These methods have the added advantage of being able to update its model and evolve the fuzzy rules with every new training instance. However, offline methods remain suitable for many real-world scenarios.

VI. Conclusion

This paper proposed the FASCOM network. It is encoded through a offline batch mode learning process consisting of three phases—one unsupervised phase followed by two supervised phases. Based on biologically inspired uniform information density, the unsupervised initialization of membership functions in the first phase led to more efficient representation and transmission of information. The supervised phase of Hebbian learning was responsible for identifying weights for fuzzy rules. The third supervised phase of error reduction not only helped fine-tune the network, but also it provided insights of how input dimensions influenced the various portions of the output. In layers 3 and 4, the representation of multiple antecedents and consequences resulted in the reduction in crosstalk for nonmutually orthogonal input patterns.

Experiments performed demonstrated that the above technical innovations led to significant improvement in the results in terms of accuracy of nonlinear estimation, despite the problem of high space complexity. The positive experimental results on real-world data (Sections V-D and V-E) lead us to believe that FASCOM could be applied to a wide variety of real-world problems for either classification (e.g., medical prognosis), regression (e.g., time-series forecasting), or multivariate analysis (e.g., financial data analysis). Future work would also involve addressing limitation of the high space complexity experienced in layer 3 and layer 4.

References

[1] C. T. Lin and C. S. Lee, *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[2] L. A. Zadeh, "Fuzzy logic, neural networks and soft computing," *Commun. ACM*, vol. 37, no. 3, pp. 77–84, 1994.

[3] J. J. Buckley and Y. Hayashi, "Fuzzy neural networks: A survey," *Fuzzy Sets Syst.*, vol. 66, no. 1, pp. 1–13, 1994.

[4] J. J. Buckley and Y. Hayashi, "Neural nets for fuzzy systems," *Fuzzy Sets Syst.*, vol. 71, no. 3, pp. 265–276, 1995.

[5] L. A. Zadeh, "Fuzzy sets," *Inf. Control*, vol. 8, pp. 338–353, 1965.

[6] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *Int. J. Man-Mach. Studies*, vol. 7, no. 1, pp. 1–13, 1975.

[7] T. Takagi and M. Sugeno, "Derivation of fuzzy control rules from human operator's control actions," in *Proc. IFAC Symp. Fuzzy Inf. Knowl. Represent. Decision Anal.*, 1983, pp. 55–60.

[8] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modelling and control," *IEEE Trans. Syst. Man Cybern.*, vol. 15, no. 1, pp. 116–132, Feb. 1985.

[9] M. Sugeno and G. T. Kang, "Structure identification of fuzzy model," *Fuzzy Sets Syst.*, vol. 28, pp. 15–33, 1988.

[10] S. Guillaume, "Designing fuzzy inference systems from data: An interpretability-oriented review," *IEEE Trans. Fuzzy Syst.*, vol. 9, no. 3, pp. 426–443, Jun. 2001.

[11] J. Casillas, O. Cordón, F. Herrera, and L. Magdalena, *Interpretability Issues in Fuzzy Modeling*. Berlin, Germany: Springer-Verlag, 2003.

[12] K. K. Ang and C. Quek, "RSPOP: Rough set-based pseudo outer-product fuzzy rule identification algorithm," *Neural Comput.*, vol. 17, pp. 205–243, 2005.

[13] B. Kosko, *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*. Englewood Ciffs, NJ: Prentice-Hall, 1992.

[14] C. T. Lin and C. S. Lee, "Neural-network-based fuzzy logic control and decision system," *IEEE Trans. Comput.*, vol. 40, no. 12, pp. 1320–1336, Dec. 1991.

[15] I. Hayashi, H. Nomura, H. Yamasaki, and N. Wakami, "Construction of fuzzy inference rules by NDF and NDFL," *Int. J. Approx. Reason.*, vol. 6, no. 2, pp. 241–266, 1992.

[16] R. R. Yager, "Modeling and formulating fuzzy knowledge bases using neural networks," *Neural Netw.*, vol. 7, no. 8, pp. 1273–1283, 1994.

[17] C. Quek and W. L. Tung, "A novel approach to the derivation of fuzzy membership functions using the Falcon-MART architecture," *Pattern Recognit. Lett.*, vol. 22, no. 9, pp. 941–958, 2001.

[18] M. Lee, S. Y. Lee, and C. H. Park, "A new neuro-fuzzy identification model of nonlinear dynamic systems," *Int. J. Approx. Reason.*, vol. 10, pp. 30–44, 1994.

[19] H. Ishibuchi, H. Tanaka, and H. Okada, "Interpolation of fuzzy if-then rules by neural networks," *Int. J. Approx. Reason.*, vol. 10, pp. 3–27, 1994.

[20] P. Werbos, "Backpropagation: Past and future," in *Proc. 2nd Int. Conf. Neural Netw.*, 1988, pp. 343–353.

[21] V. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.

[22] J. S. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst. Man Cybern.*, vol. 23, no. 3, pp. 665–685, May/Jun. 1993.

[23] N. Kasabov, "Evolving fuzzy neural networks for supervised/unsupervised online," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 31, no. 6, pp. 902–918, Dec. 2001.

[24] N. Kasabov and Q. Song, "DENFIS: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 2, pp. 144–154, Apr. 2002.

[25] P. K. Simpson, "Fuzzy min-max neural networks-Part 1: Classification," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 776–786, Sep. 1992.

[26] S. Abe and M.-S. Lan, "A method for fuzzy rules extraction directly from numerical data and its application to pattern classification," *IEEE Trans. Fuzzy Syst.*, vol. 3, no. 1, pp. 18–28, Feb. 1995.

[27] A. Joshi, N. Ramakrishman, E. N. Houstis, and J. R. Rice, "On neurobiological, neuro-fuzzy, machine learning, and statistical pattern recognition techniques," *IEEE Trans. Neural Netw.*, vol. 8, no. 1, pp. 18–31, Jan. 1997.

[28] K. K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philosoph. Mag.*, vol. 2, no. 6, pp. 559–572, 1901.
[29] N. Suga, "Cortical computation maps for auditory imaging," *Neural Netw.*, vol. 3, pp. 3–21, 1990.
[30] J. Rovamo, V. Virsu, and R. Näsänen, "Cortical magnification factors predicts the photopic contrast sensitivity of peripheral vision," *Nature*, vol. 271, pp. 54–6, 1978.
[31] P. Azzopardi and A. Cowey, "The overrepresentation of the fovea and adjacent retina in the striate cortex and dorsal lateral geniculate nucleus of the macaque monkey," *Neuroscience*, vol. 72, pp. 627–639, 1996.
[32] S. W. Kuffler, J. G. Nicholls, and A. R. Martin, *From Neuron to Brain: A Cellular Approach to the Function of the Nervous System*, 2nd ed. Sunderland, MA: Sinauer Associates, 1984.
[33] A. Das, "Plasticity in adult sensory cortex: A review," *Network: Comput. Neural Syst.*, vol. 8, no. 2, pp. R33–R76, 1997.
[34] M. D. Plumbley, "Do cortical maps adapt to optimize information density?," *Network: Comput. Neural Syst.*, vol. 10, pp. 41–58, 1999.
[35] H. Nakanishi, I. B. Turksen, and M. Sugeno, "A review and comparison of six reasoning methods," *Fuzzy Sets Syst.*, vol. 57, no. 3, pp. 257–294, 1993.
[36] E. R. Kandel, J. H. Schwartz, and T. M. Jessell, *Principles of Neural Science*, 4th ed. New York: McGraw-Hill, 2000.
[37] F. Junbo, J. Fan, and S. Yan, "A learning rule for fuzzy associative memories," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 1994, pp. 4273–4277.
[38] F. Chung and T. Lee, "On fuzzy associative memory with multiple-rule storage capacity," *IEEE Trans. Fuzzy Syst.*, vol. 4, no. 3, pp. 375–384, Aug. 1996.
[39] P. Liu, "The fuzzy associative memory of max-min fuzzy neural networks with threshold," *Fuzzy Sets Syst.*, vol. 107, pp. 147–157, 1999.
[40] P. Sussner and M. E. Valle, "Implicative fuzzy associative memories," *IEEE Trans. Fuzzy Syst.*, vol. 14, no. 6, pp. 793–807, Dec. 2006.
[41] D. Marr and E. Hilderth, "The theory of edge detection," in *Proc. R. Soc. London B*, 1980, vol. 207, pp. 187–217.
[42] C. Enroth-Cugell and J. Robson, "The contrast sensitivity of retinal ganglion cells of the cat," *J. Physiol.*, vol. 187, pp. 517–522, 1966.
[43] J. Yen and R. Langari, *Fuzzy Logic: Intelligence, Control, and Information*. Englewood Cliffs, NJ: Prentice-Hall, 1999.
[44] G. J. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic, Theory and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1995.
[45] C. Quek and R. W. Zhou, "The POP learning algorithms: Reducing work in identifying fuzzy rules," *Neural Netw.*, vol. 14, no. 10, pp. 1431–1445, 2001.
[46] N. V. Swindale, "How different feature spaces may be represented in cortical maps," *Network: Comput. Neural Syst.*, vol. 15, pp. 217–242, 2004.
[47] C. Koch, "Passive denritic trees," in *Biophysics of Computation: Information Processing in Single Neurons*. New York: Oxford Univ. Press, 1999, pp. 49–84.
[48] S.-G. Kong and B. Kosko, "Adaptive fuzzy systems for backing up a truck-and-trailer," *IEEE Trans. Neural Netw.*, vol. 3, no. 2, pp. 211–223, Mar. 1992.
[49] D. O. Hebb, *The Organization of Behavior; A Neuropsychological Theory*. New York: Wiley, 1949.
[50] L. R. Squire and E. R. Kandel, *Memory : From Mind to Molecules*. New York: Scientific American Library, 2000.
[51] K. J. Lang and M. J. Witbrock, "Learning to tell two spirals apart," in *Proc. Connectionist Models Summer School*, 1988, pp. 52–59.
[52] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugenics, Part II*, vol. 7, pp. 179–188, 1936.
[53] A. Singh, C. Quek, and S.-Y. Cho, "DCT-Yager FNN: A novel Yager-based fuzzy neural network with the discrete clustering technique," *IEEE Trans. Neural Netw.*, vol. 19, no. 4, pp. 625–644, Apr. 2008.
[54] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 698–713, Sep. 1992.
[55] W. L. Tung and C. Quek, "GenSoFNN: A generic self-organizing fuzzy neural network," *IEEE Trans. Neural Netw.*, vol. 13, no. 5, pp. 1075–1086, Sep. 2002.
[56] G. S. Ng, C. Quek, and H. Jian, "FCMAC-EWS: A bank failure early warning system based on a novel localized pattern learning and semantically associative fuzzy neural network," *Expert Syst. Appl.*, vol. 34, no. 2, pp. 989–1003, 2008.
[57] Z. Guo, C. Quek, and D. Maskell, "FCMAC-AARS: A novel FNN architecture for stock market prediction and trading," in *Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, 2006, pp. 2375–2381.
[58] J. Sim, W. L. Tung, and C. Quek, "FCMAC-Yager: A novel Yager inference scheme based fuzzy CMAC," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1394–1410, Nov. 2006.
[59] C. W. Ting and C. Quek, "A novel blood glucose regulation using TSK0-FCMAC: A fuzzy CMAC based on the zero-ordered TSK fuzzy inference scheme," *IEEE Trans. Neural Netw.*, vol. 20, no. 5, pp. 856–871, May 2009.
[60] K. H. Quah and C. Quek, "FITSK: Online local learning with generic fuzzy input Takagi-Sugeno-Kang fuzzy framework for nonlinear system estimation," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 36, no. 1, pp. 166–178, Feb. 2006.
[61] F. Liu, C. Quek, and G. S. Ng, "A novel generic Hebbian ordering-based fuzzy rule base reduction approach to Mamdani neuro-fuzzy system," *Neural Comput.*, vol. 19, pp. 1656–1680, 2007.
[62] K. K. Ang, C. Quek, and M. Pasquier, "POPFNN-CRI(S): Pseudo outer product-based fuzzy neural network using the compositional rule of inference and singleton fuzzifier," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 33, no. 6, pp. 838–849, Dec. 2003.
[63] S. L. Chiu, "Fuzzy model identification based on cluster estimation," *J. Intell. Fuzzy Syst.*, vol. 2, no. 3, pp. 267–278, 1994.
[64] E. H. Mamdani and S. Assilian, "Applications of fuzzy algorithm for control of a simple dynamic plant," *Proc. IEEE*, vol. 121, no. 12, pp. 1585–1588, Dec. 1974.
[65] I. B. Turksen, "Interval valued fuzzy sets based on normal forms," *Fuzzy Sets Syst.*, vol. 20, pp. 191–210, 1986.
[66] K. K. Ang and C. Quek, "Stock trading using RSPOP: A novel rough set based neuro-fuzzy approach," *IEEE Trans. Neural Netw.*, vol. 17, no. 5, pp. 1301–1315, Sep. 2006.
[67] A. Tan, C. Quek, and K. Yow, "Maximizing winning trades using a rough set based other-product (RSPOP) fuzzy neural network intelligent stock trading system," *Appl. Intell.*, vol. 29, no. 2, pp. 116–128, 2008.
[68] G. K. Tan, "Feasibility of predicting congestion states with neural networks," Nanyang Technological Univ., Singapore, Tech. Rep., 1997.

Hanlin Goh received the B.Eng. degree in computer engineering and the M.Sc. degree in bioinformatics from Nanyang Technological University, Singapore, in 2007 and 2009, respectively.

He is currently a Research Officer with the Computer Vision and Image Understanding Department, Institute for Infocomm Research, Singapore, and a member of Image Perception, Access and Language (IPAL), a French-Singaporean joint laboratory. His research interests include fuzzy neural networks, machine learning, computer vision, and image processing.

Joo-Hwee Lim (M'07) received the B.Sc. (Hons I) and M.Sc. (by research) degrees in computer science from the National University of Singapore, Singapore, in 1989 and 1991, respectively, and the Ph.D. degree in computer science and engineering from the University of New South Wales, Sydney, Australia, in 2004.

He joined Institute for Infocomm Research (I2R) and its predecessors, Singapore in October 1990. He is currently the Department Head of the Computer Vision & Image Understanding Department, with staff strength of 50 research scientists and engineers, at I2R, Singapore. He is also the Co-Director of Image Perception, Access and Language (IPAL), a French-Singapore Joint Lab (UMI 2955, January 2007–December 2010) and was bestowed the title of "Chevallet dans l'ordre des Palmes Academiques" by the French Government in 2008. He has conducted research in connectionist expert systems, neural-fuzzy systems, handwriting recognition, multiagent systems, and content-based retrieval. He has ten patents (awarded and pending) and published more than 120 refereed international journal and conference papers in his research areas.

Chai Quek (M'96) received the B.Sc. degree in electrical and electronics engineering and the Ph.D. degree in intelligent control from Heriot-Watt University, Edinburgh, Scotland, in 1986 and 1990, respectively.

He is an Associate Professor and a member of the Centre for Computational Intelligence, formerly the Intelligent Systems Laboratory, School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include intelligent control, neurocognitive architectures, artificial intelligence (AI) in education, neural networks, fuzzy systems, fuzzy-rule-based systems, and genetic algorithms and brain-inspired neurocognitive applications in computational finance and biomedical engineering.

Dr. Quek is a member of the IEEE Technical Committee on Computational Finance.